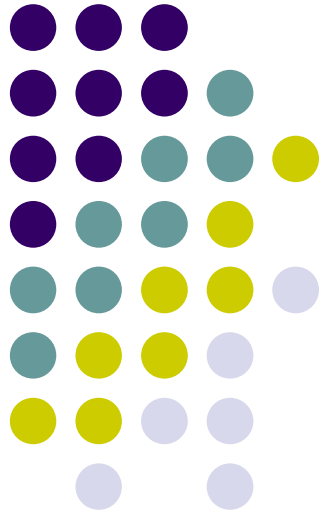
# ME964
# GIT, Trac, CMake & CUDA Debugging

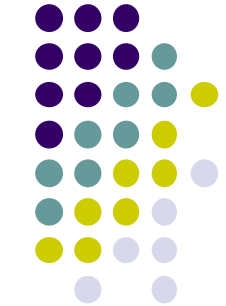


Spring 2012

Brian J. Davis
PhD. Candidate Biomedical Engineering
Research Assistant
School of Medicine and Public Health
Research Area Medical Physics
bdavis5@wisc.edu

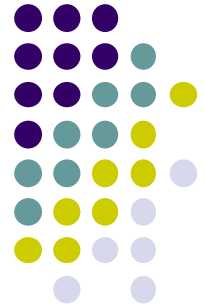
© Dan Negrut, 2012
ME964 UW-Madison
© Brian J. Davis 2012

# What this talk is about… What I hope you to gain from it



- Your armed with the knowledge to program a GPU… now what?
- How to manage, track, build, and debug your CUDA/GPU project.
- Using tools like:
  - GIT - source control management / version control software
  - Trac - bug tracking and documentation/wiki
  - CMake – cross platform build specification, test case, and packaging (release) tool
  - Nsight/cuda-gdb CUDA debugging
- Though I may not go in depth hopefully this will provide at least some exposure to the tools that are out there

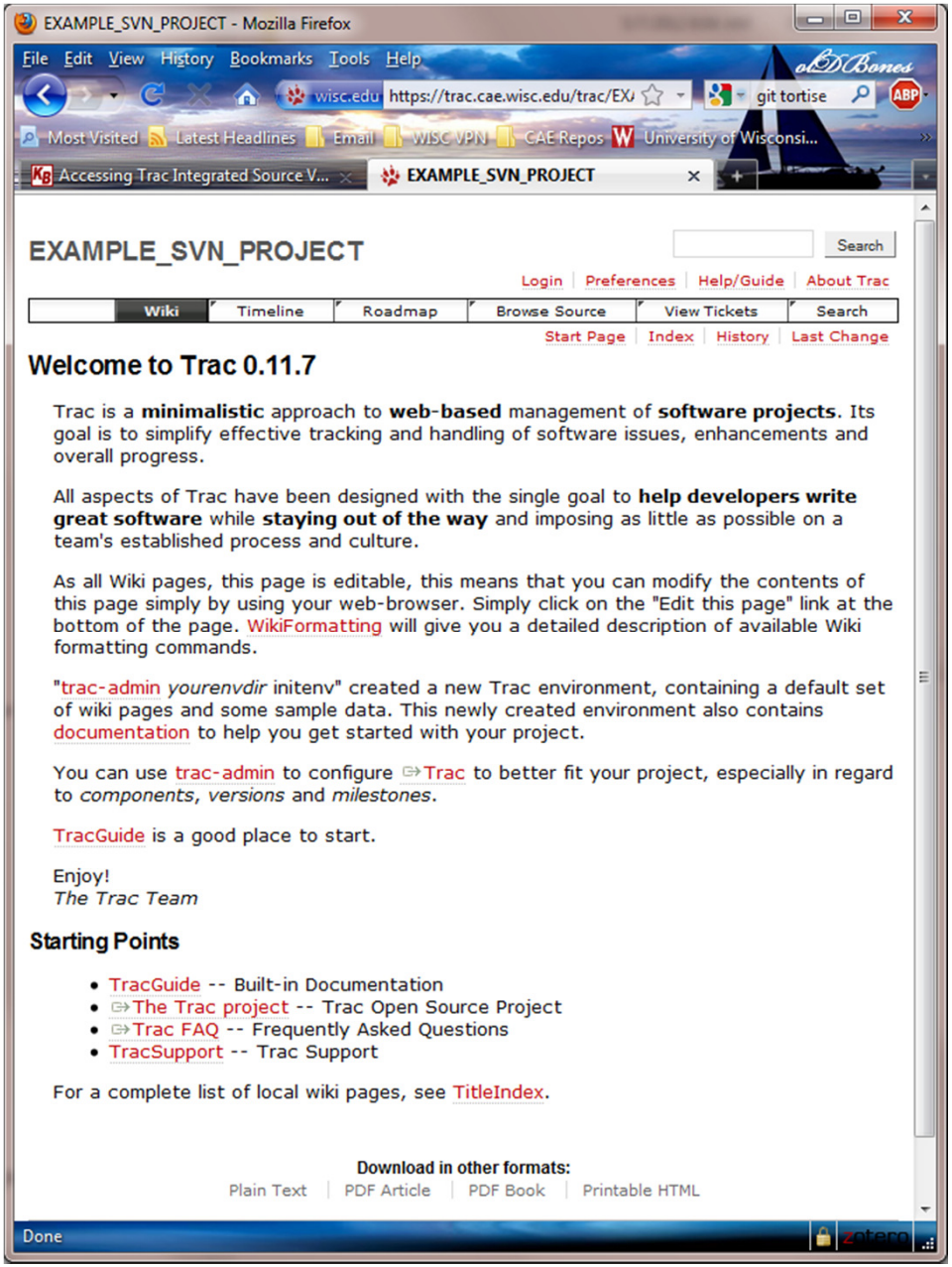
# Source Control Management (SCM)



- We have resources at CAE… Use them!
- CAE resources
  - https://kb.wisc.edu/cae/page.php?id=22527
- Trac bug tracking with web based source browsing also supported
- CAE provides: **Git**, HG/Mercurial, Bazarr, and SVN (if you have to)

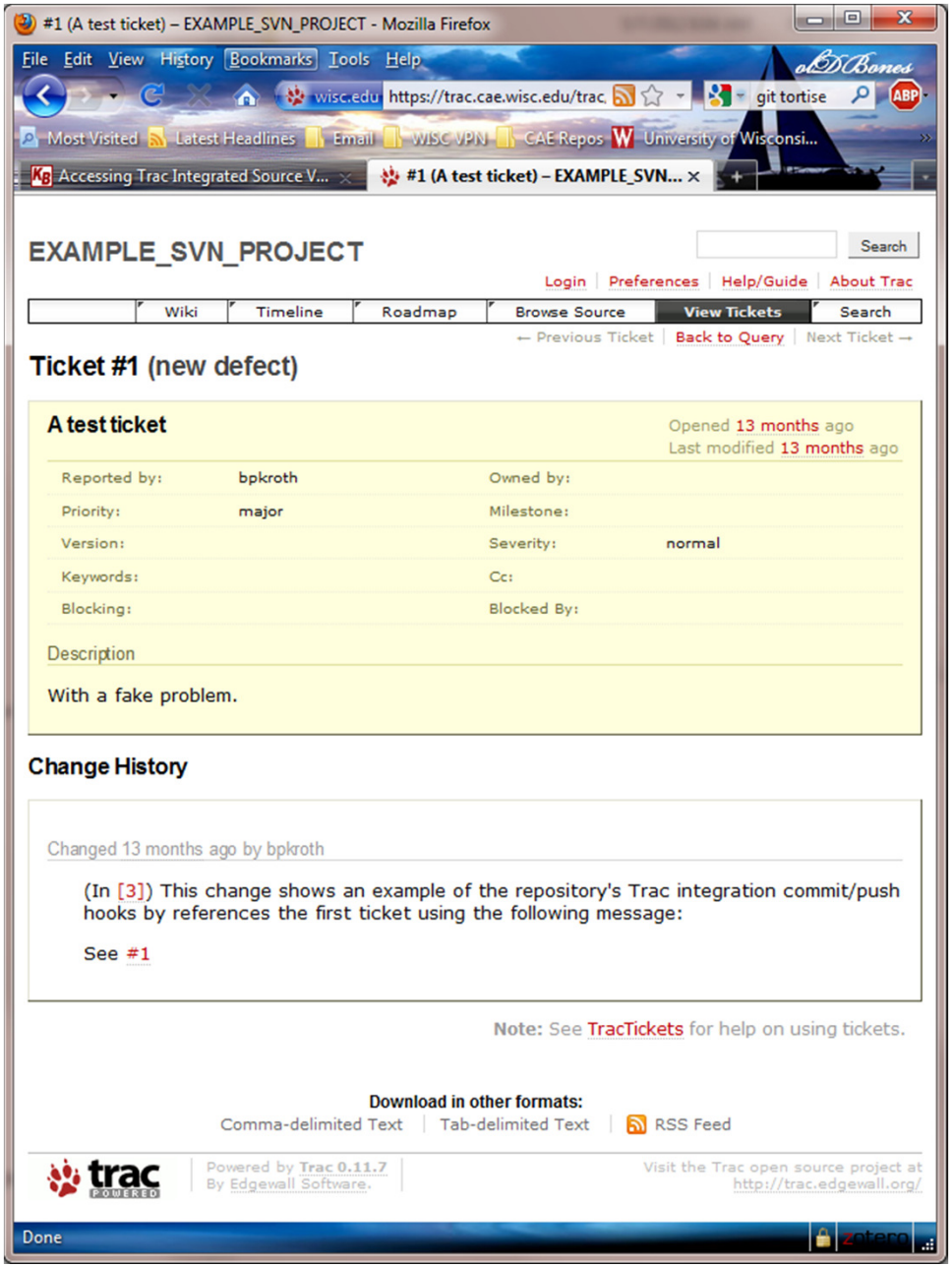
# Trac – wiki, bug, and feature tracking

- Wiki – allows editing of project documentation through web browser


EXAMPLE_SVN_PROJECT - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Most Visited Latest Headlines Email WISC VPN CAE Repos University of Wisconsi...
Accessing Trac Integrated Source V...
EXAMPLE_SVN_PROJECT
EXAMPLE_SVN_PROJECT
Search
Login | Preferences | Help/Guide | About Trac
Wiki | Timeline | Roadmap | Browse Source | View Tickets | Search
Start Page | Index | History | Last Change
Welcome to Trac 0.11.7
Trac is a minimalistic approach to web-based management of software projects. Its goal is to simplify effective tracking and handling of software issues, enhancements and overall progress.
All aspects of Trac have been designed with the single goal to help developers write great software while staying out of the way and imposing as little as possible on a team's established process and culture.
As all Wiki pages, this page is editable, this means that you can modify the contents of this page simply by using your web-browser. Simply click on the "Edit this page" link at the bottom of the page. WikiFormatting will give you a detailed description of available Wiki formatting commands.
"trac-admin yourenvdir initenv" created a new Trac environment, containing a default set of wiki pages and some sample data. This newly created environment also contains documentation to help you get started with your project.
You can use trac-admin to configure Trac to better fit your project, especially in regard to components, versions and milestones.
TracGuide is a good place to start.
Enjoy!
The Trac Team
Starting Points
TracGuide -- Built-in Documentation
The Trac project -- Trac Open Source Project
Trac FAQ -- Frequently Asked Questions
TracSupport -- Trac Support
For a complete list of local wiki pages, see TitleIndex.
Download in other formats:
Plain Text | PDF Article | PDF Book | Printable HTML
Done

# Trac – wiki, bug, and feature tracking

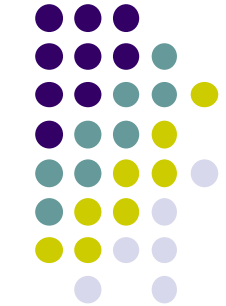
- Bug Tracking


#1 (A test ticket) – EXAMPLE_SVN_PROJECT - Mozilla Firefox
File Edit View History Bookmarks Tools Help
wisc.edu https://trac.cae.wisc.edu/trac
git tortise
Most Visited Latest Headlines Email WISC VPN CAE Repos University of Wisconsi...
Accessing Trac Integrated Source V...
#1 (A test ticket) – EXAMPLE_SVN...
EXAMPLE_SVN_PROJECT
Search
Login Preferences Help/Guide About Trac
Wiki Timeline Roadmap Browse Source View Tickets Search
← Previous Ticket Back to Query Next Ticket →
Ticket #1 (new defect)
A test ticket
Opened 13 months ago
Last modified 13 months ago
Reported by: bpkroth
Owned by:
Priority: major
Milestone:
Version:
Severity: normal
Keywords:
Cc:
Blocking:
Blocked By:
Description
With a fake problem.
Change History
Changed 13 months ago by bpkroth
(In [3]) This change shows an example of the repository's Trac integration commit/push hooks by references the first ticket using the following message:
See #1
Note: See TracTickets for help on using tickets.
Download in other formats:
Comma-delimited Text Tab-delimited Text RSS Feed
trac
Powered by Trac 0.11.7
By Edgewall Software.
Visit the Trac open source project at
http://trac.edgewall.org/
Done

# Trac – wiki, bug, and feature tracking

- Online source browsing

# Not all SCM tools are equal

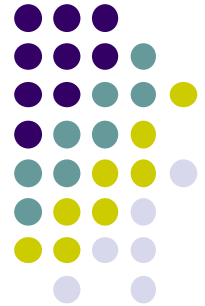


- Centralized (SVN, CVS) vs. Distributed (GIT)
  - With distributed you work offline such as on plane, sailboat, or a cabin in the woods
  - Create a copy for a friend without ever having a remote server
- Deep Cloning
- Performance

# Not all SCM tools are equal

- What you check in to the repository is what you get out.
- Unbelievably not all tools support this as corruption on drives, backups, or transfer to CD/DVD can cause data corruption and no integrity checking is performed.

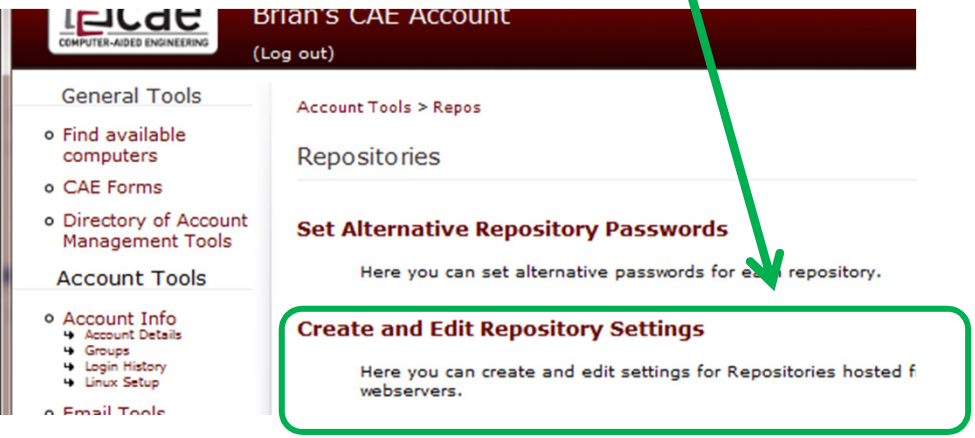
# GIT

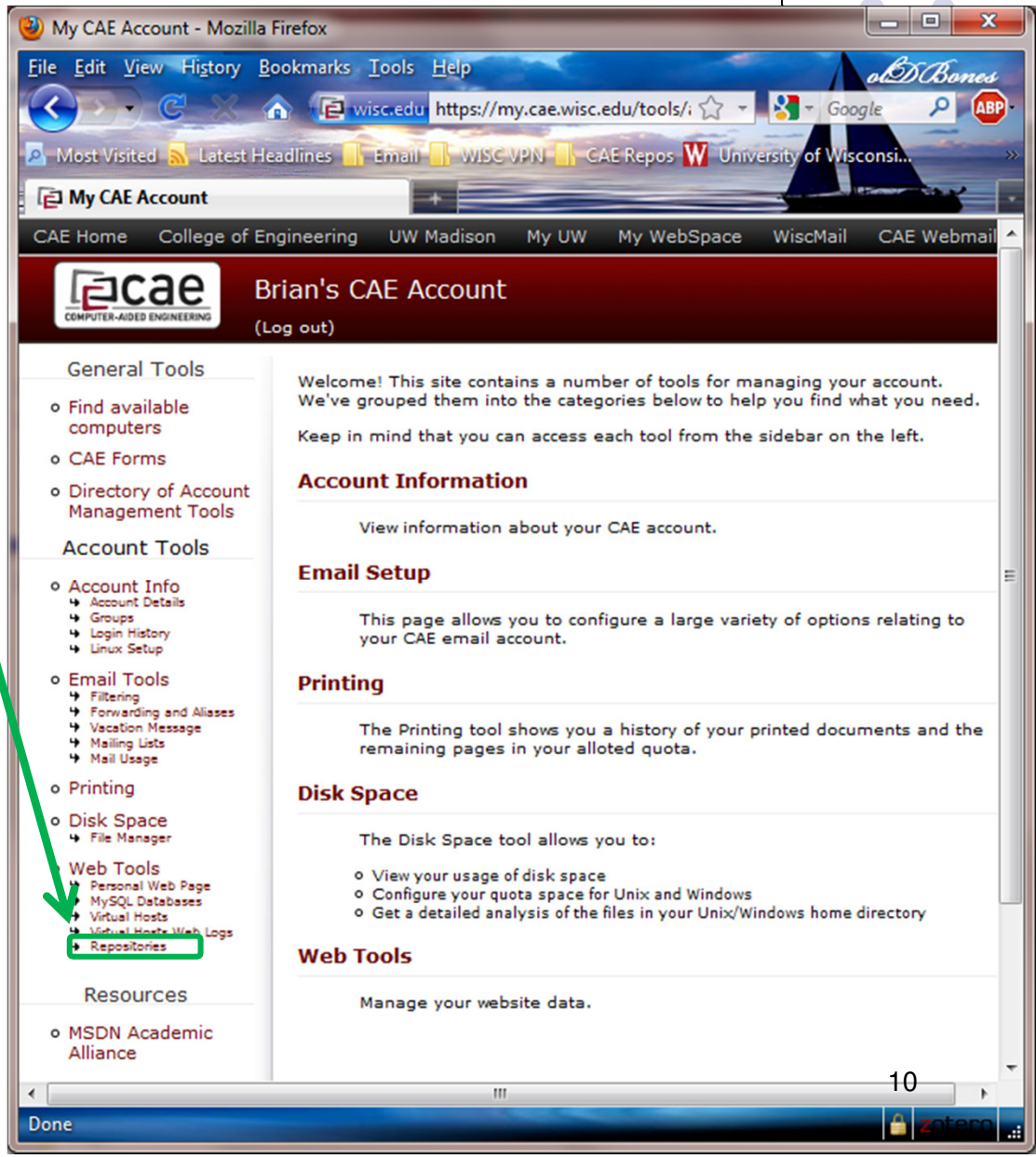


- http://git-scm.com/
- **Tech *Talk*: Linus Torvalds on *git***
  - **http://www.youtube.com/watch?v=4XpnKHJAok8**
- **Tortise Gui for GIT**
  - **http://code.google.com/p/tortoisegit/**
- **GIT Bash for Windows**
  - http://msysgit.github.com/

# A brief GIT How To

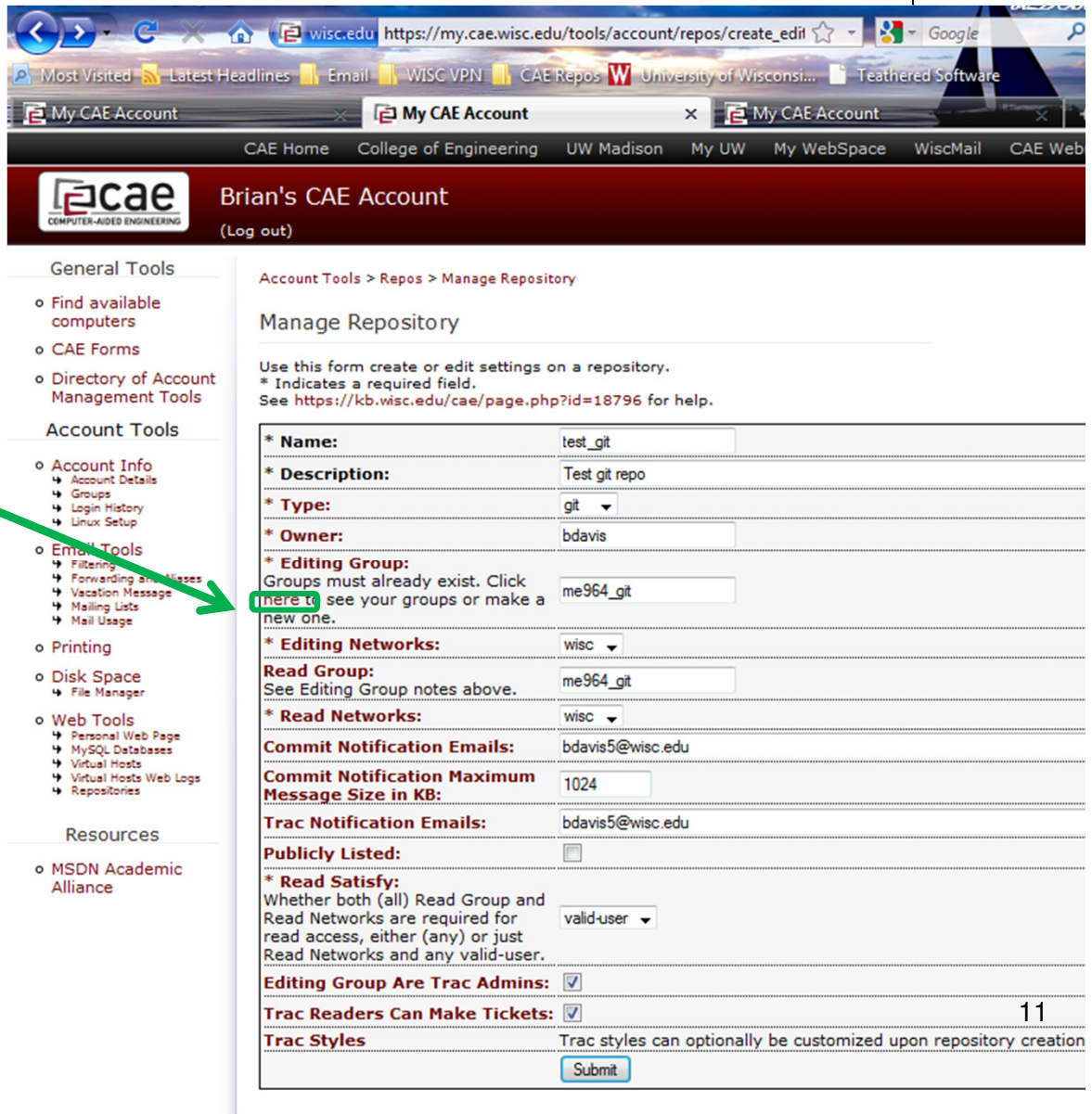
- Login to CAE account
- Click Repositories under Web Tools
- Click Create and Edit Repository Settings



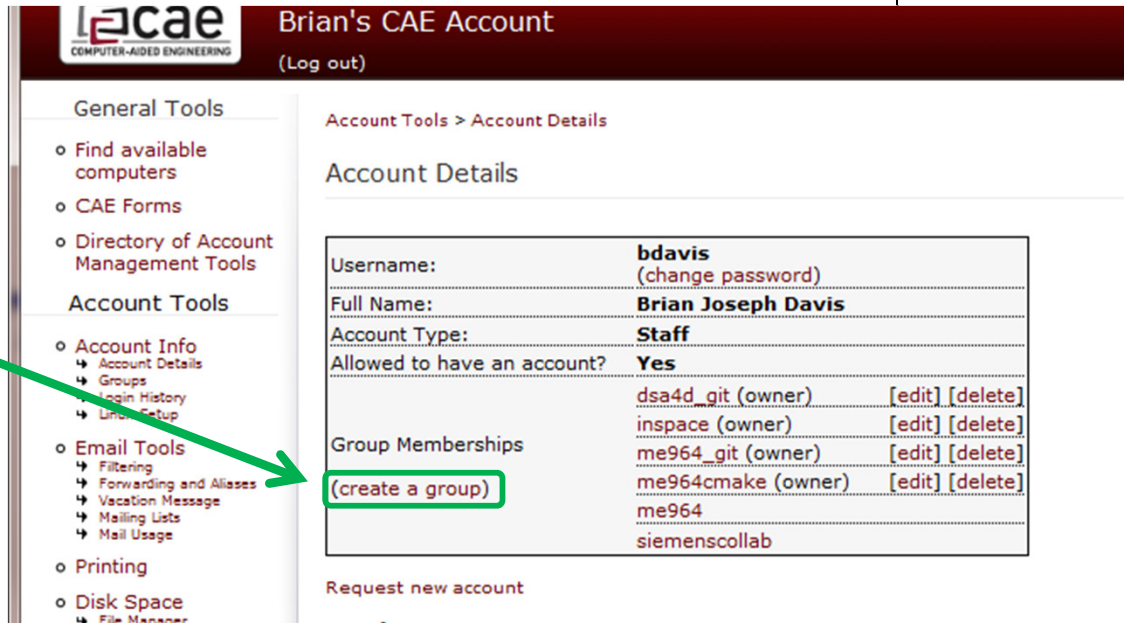
# A brief GIT How To

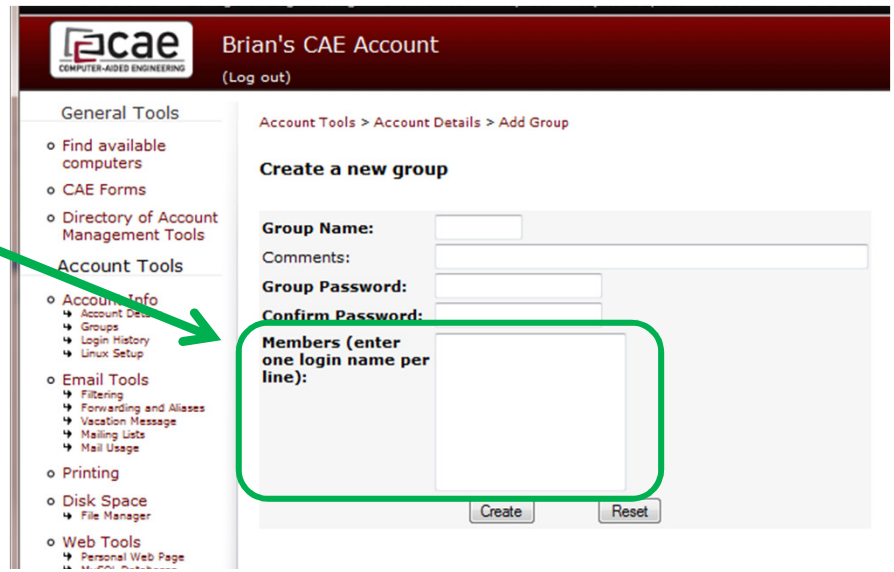
- Give the repo a name
- Set editing group if needed



# A brief GIT How To

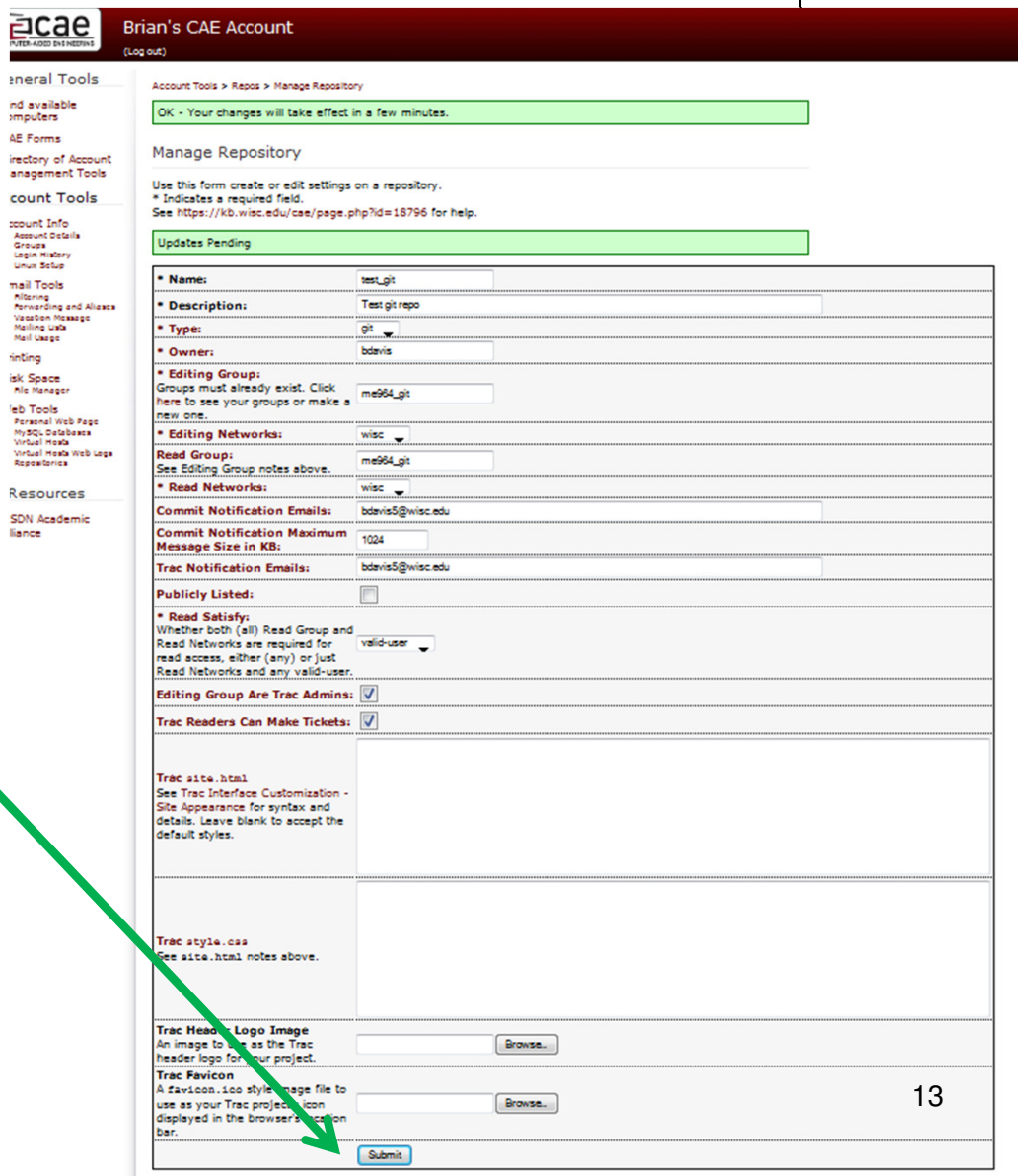
- Create an editing group to allow others with cae logins to access your repository
- Add group members



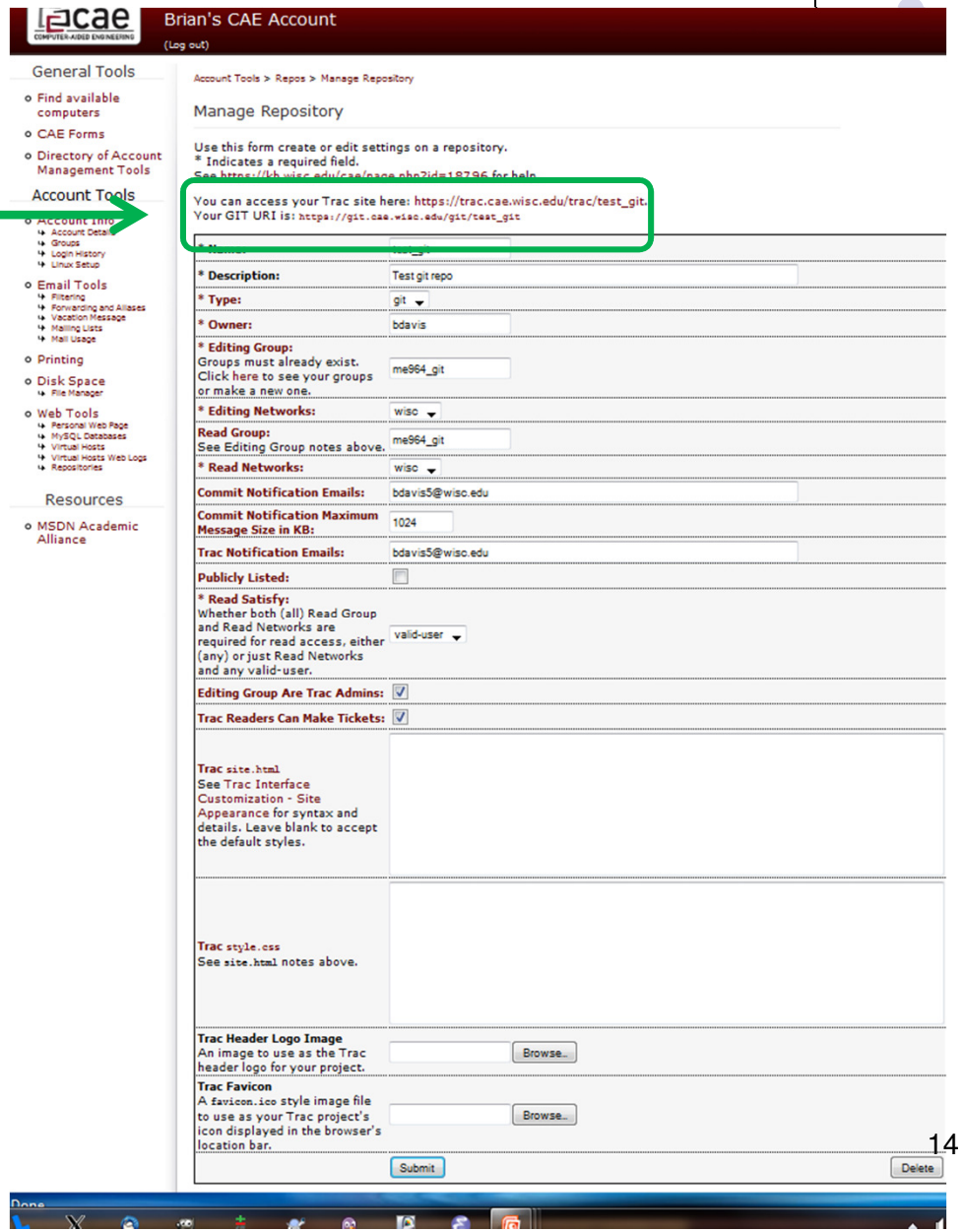
# A brief GIT How TO

- Trac settings can be modified if desired
- Click submit



# A brief GIT How To

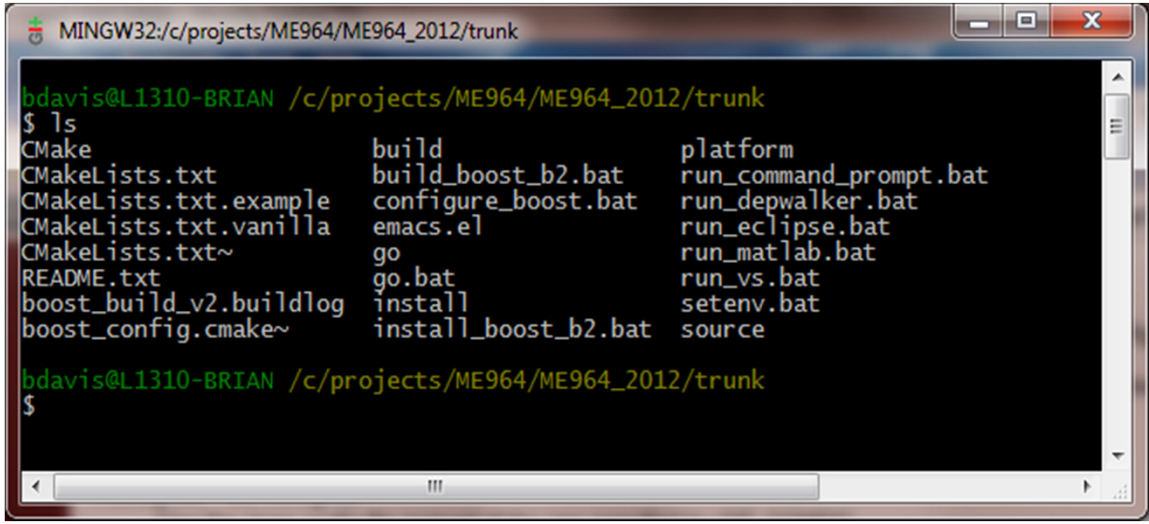
- Trac and GIT urls will then be listed



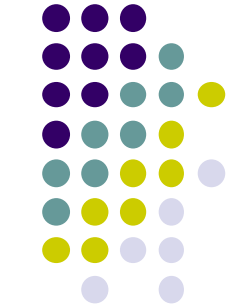
# A brief GIT How To

- GIT bash will be used for this example.
  - http://msysgit.github.com/
- Create a directory for your source
  - Preferably a short path
  - Without spaces
- I used

  C:\projects\ME964\ME964_2012\trunk\



# A brief GIT How To



- Initialize a GIT repository
- A .git directory will be created
- Set your username and email address
- --global can be used to set for all future repositories you create on the computer you are using – it gets set in your profile

```
cd /c/projects/ME964/ME964_2012/trunk

git init

git config --global user.name "Brian J. Davis“

git config --global user.email "bdavis5@wisc.edu"
```

# A brief GIT How To

- Add your files to git
- Commit the project to the local repository –a for all files –m for a message
- Create a bare clone of the repo
- List the config and look at remote.origin.url=c:/projects/ME964/ME964_2012/trunk/.git
- Change the remote url to point to the remote cae repository
- Push your code to the remote repository

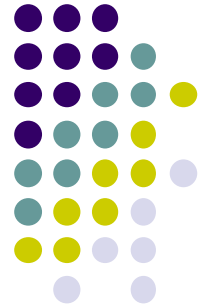
```
git add CMakeLists.txt

git add my_dir_1

git add my_dir/*.cpp

 git commit -a -m "creating project“

#back out of the local repo
cd ../
git clone --bare trunk/.git  me964.git
git config –l

git config remote.origin.url
https://git.cae.wisc.edu/git/me964_git

$ git push origin master
Username:
Password:
Counting objects: 135, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (119/119), done.
Writing objects:  36% (49/135)
```

# A brief GIT How To

- Clone the remote repo to a new local copy
- Make changes
- Commit changes
- Push changes to remote repository

```
git clone https://git.cae.wisc.edu/git/me964_git  my_checkout

# make changes …. Make changes…
git commit –a –m ‘my message’

# make more changes …. Make more changes…
git commit –a –m ‘my more changes message’

git push origin master
```

# CMake



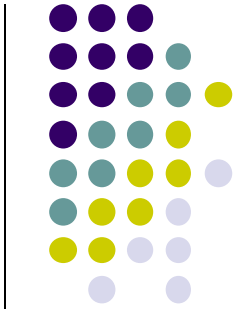
- Build tools – What do they do?
- What is CMake?
- Why use CMake?
- What can CMake do?
- Example Project
- Don't listen to me go to the source… hey man I am just spreading the word.
  - Google Tech Talk
    - CMake/CPack/CTest/CDash Open Source Tools to Build Test and Deploy C++ Software
    - http://www.youtube.com/watch?v=8Ut9o4OdSC0&feature=youtube_gdata
- I am **not affiliated** in any way with Kitware. I am just trying to get my code to compile on Microsoft Windows and Linux.

# The Source

- CMake by Kitware
  - http://www.CMake.org/
  - Documentation
    - http://www.cmake.org/cmake/help/documentation.html
    - http://www.cmake.org/cmake/help/cmake-2-8-docs.html
  - FAQ
    - http://www.CMake.org/Wiki/CMake_FAQ
  - Download
    - http://www.CMake.org/CMake/resources/software.html
  - Source Repository
    - http://www.CMake.org/Wiki/CMake/Git
  - Tutorial
    - http://www.CMake.org/CMake/help/CMake_tutorial.html

"Go right to the source and ask the horse. He'll give you the answer that you'll endorse. He's always on a steady course. Talk to Mr. Ed" – Theme song to Mr. Ed, 1961-1966

# Build tools what do they do?



- Check dependencies
  - You changed a source file and the .obj file needs to be rebuilt which then rebuilds the dll or exe.
  - Controls what gets built in what order
- Execute generators
  - Such as the compiler which coverts source to output files
  - Generate configuration files
  - Specify install and testing locations and put all files in correct locations.
  - Generate code
    - Lexx Yacc or Flex and Bison
    - Extended Backus–Naur Form (EBNF) – context -free grammars
    - SWIG C/C++/C# (Mono) Language integration in Linux
- Check system for build dependencies (some do VS not so much)
  - CUDA, nvcc,
    - Object files to be created from source
  - Library dependencies (VTK, ITK, etc)
  - What are capabilities/versions of C and C++ stdlib etc.
- Whatever you command them to do

```
MESSAGE( “Executing world domination script now” )
execute_process(
        COMMAND world_domination.bat
        WORKING_DIRECTORY ./world_domination_scripts/milky_way_galaxy/earth
        INPUT_FILE world_domination.cfg
        OUTPUT_FILE world_domination_result.log
        ERROR_FILE  world_domination_error.log
)
```

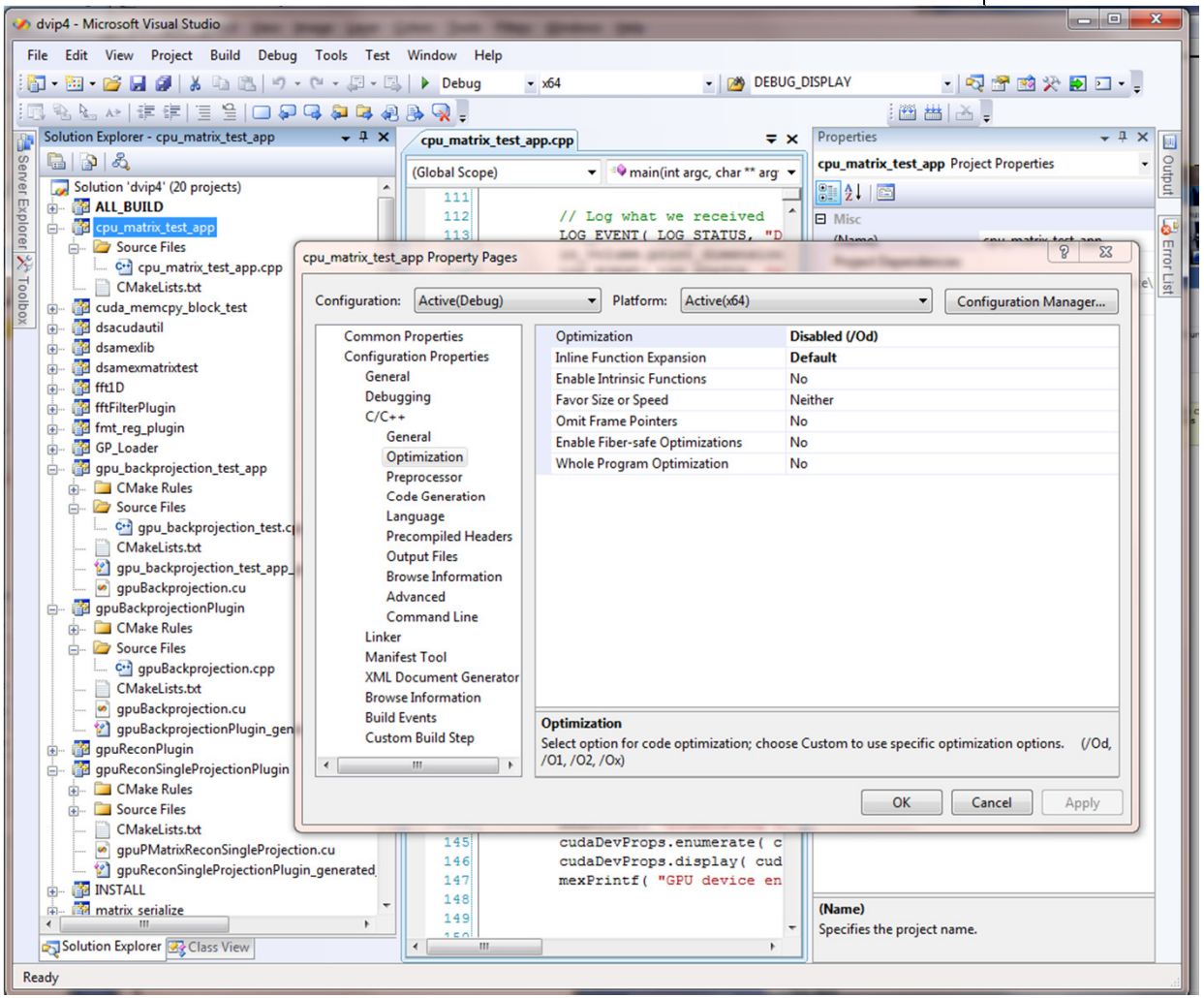
# Build Tools

- What have I used? What comparisons can I make?
  - BorlandBuilder 5.02/Builder C++ 4, and 6
  - MS VisualStudio
  - GNU Make, autoconf, and friends
  - BJam – BoostBuildV2 part of Boost C++
    - http://www.boost.org/boost-build2/
    - Perforce Software (not related to Boost) Jam Tutorial – only put here due to use of Jam
      - http://www.perforce.com/perforce/conferences/us/2001/wingerd/WPLaura.pdf
      - I have not used the Perforce version only Boost's version
  - CMake

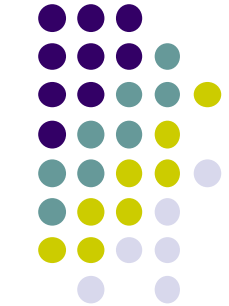
# Visual Studio as a build tool

**>>> MY OPINIONS <<<**

- Based on What? – ~10 years experience
- Great for simple projects
- Unwieldy for complex projects
- Modal Dialog boxes that can't be resized?
- Build Spec is not searchable
- Which spec am I changing? all, debug, release? Is this for 32 or x64?
- What did I change that made the build break? I know I can diff the .sln and .prj files. We will take a look at those (next slide)
- All seems hidden behind GUI which is difficult access, change, and maintain . Especially with broad sweeping changes. Ok there is project inheritance.
- There is also nmake as an alternative.
- Good for wear leveling of your mouse buttons through increased use of right click

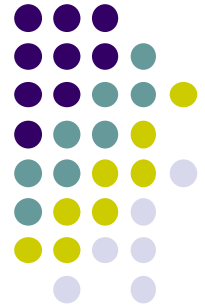


# Looking at (diffing) VS project files WHEN things go wrong: .sln



```
Microsoft Visual Studio Solution File, Format Version 10.00
# Visual Studio 2008
Project("{8BC9CEB8-8B4A-11D0-8D11-00A0C91BC942}") = "CudaCollision", "CudaCollision\CudaCollision.vcproj", "{08C6F311-7AA6-46EB-BFB9-7F947F5DD014}"
EndProject
Project("{8BC9CEB8-8B4A-11D0-8D11-00A0C91BC942}") = "BulletValidation", "BulletValidation\BulletValidation.vcproj", "{8644F016-E5EF-432D-98C6-91D27D459746}"
EndProject
Project("{8BC9CEB8-8B4A-11D0-8D11-00A0C91BC942}") = "DataGeneration", "DataGeneration\DataGeneration.vcproj", "{A0B424D7-6CCD-465C-A90E-D5589E73954E}"
EndProject
Global
        GlobalSection(SolutionConfigurationPlatforms) = preSolution
                Debug|Win32 = Debug|Win32
                Debug|x64 = Debug|x64
                Release|Win32 = Release|Win32
                Release|x64 = Release|x64
        EndGlobalSection
        GlobalSection(ProjectConfigurationPlatforms) = postSolution
                {08C6F311-7AA6-46EB-BFB9-7F947F5DD014}.Debug|Win32.ActiveCfg = Debug|Win32
                {08C6F311-7AA6-46EB-BFB9-7F947F5DD014}.Debug|Win32.Build.0 = Debug|Win32
                {08C6F311-7AA6-46EB-BFB9-7F947F5DD014}.Debug|x64.ActiveCfg = Debug|x64
                {08C6F311-7AA6-46EB-BFB9-7F947F5DD014}.Debug|x64.Build.0 = Debug|x64
                {08C6F311-7AA6-46EB-BFB9-7F947F5DD014}.Release|Win32.ActiveCfg = Release|Win32
                …
                …
                …
                {A0B424D7-6CCD-465C-A90E-D5589E73954E}.Release|Win32.Build.0 = Release|Win32
                {A0B424D7-6CCD-465C-A90E-D5589E73954E}.Release|x64.ActiveCfg = Release|x64
                {A0B424D7-6CCD-465C-A90E-D5589E73954E}.Release|x64.Build.0 = Release|x64
        EndGlobalSection
        GlobalSection(SolutionProperties) = preSolution
                HideSolutionNode = FALSE
        EndGlobalSection
EndGlobal
```

# Looking at (diffing) VS project files WHEN things go wrong: .sln



```
<?xml version="1.0" encoding="Windows-1252"?>
<VisualStudioProject
	ProjectType="Visual C++"
	Version="9.00"
	Name="CudaCollision"
	ProjectGUID="{08C6F311-7AA6-46EB-BFB9-7F947F5DD014}"
	RootNamespace="CudaCollision"
	Keyword="x64Proj"
	TargetFrameworkVersion="0"
	>
	<Platforms>
		<Platform
			Name="x64"
		/>
		<Platform
			Name="Win32"
		/>
	</Platforms>
	<ToolFiles>
		<DefaultToolFile
			FileName="NvCudaRuntimeApi.v3.2.rules"
		/>
	…
	… Blah Blah Blah GOBS and GOBS more where this came from
	…
	 <File
		RelativePath=".\cuda_timer.cu"
		>
	</File>
	</Files>
	<Globals>
	</Globals>
</VisualStudioProject>
```

# What does VS do?

- Ok well .prj is better than .sln not that I don't like looking at hash codes or anything.
- I counted 698 lines (ok well SciTE http://www.scintilla.org/SciTE.html line numbering did) of good wholesome xml
- Seriously 698 lines of xml to specify 1 executable (CollisionDetection) to generate a handfull of command lines????
- Who's selling hardrives these days… I need to invest!
- The point here is that this is the text output which can be viewed and diffed with diff tools and should be preferably clean and it should be a relatively short time to figure out what changed and broke the build.

# What is CMake?

- From http://www.CMake.org/
  - “Welcome to **CMake**, the cross-platform, open-source build system. CMake is a **family of tools** designed to **build**, **test** and **package** software. CMake is used to control the software compilation process using simple platform and compiler independent configuration files. CMake generates native makefiles and workspaces that can be used in the compiler environment of your choice. ”
  - Product of Kitware due to the need for a cross-platform build environment for Insight Segmentation and Registration Toolkit (ITK) as part of the Visible Human Project

# Meta build

- Meta build tool
  - A built tool that builds build files defined by build specifications (CMakeLists.txt files)
  - CMakeLists.txt files which generate Make (Linux and friends and Nmake on Windows) or project files (VS):
    - Borland Makefiles ,MSYS Makefiles, MinGW Makefiles, NMake Makefiles, NMake Makefiles JOM, Unix Makefiles, Visual Studio 10, Visual Studio 10 Win64, Visual Studio 6, Visual Studio 7, Visual Studio 7 .NET 2003, Visual Studio 8 2005, Visual Studio 8 2005 Win64, Visual Studio 9 2008, Visual Studio 9 2008 Win64, Watcom Wmake, CodeBlocks - MinGW Makefiles, CodeBlocks - NMake Makefiles, CodeBlocks - Unix Makefiles, Eclipse CDT4 - MinGW Makefiles, Eclipse CDT4 - NMake Makefiles, Eclipse CDT4 - Unix Makefiles
- First experience (before drinking the coolaid) was with cygwin where CMake bootstraped itself, built itself which was then used to generate the build files for ITK which were then used to build ITK using GNU Make. This resulted in a 4 stage to compile…. Awesome!

  ./bootstrap

  make

  make install

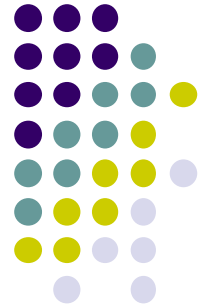
  Run CMake to generate Makefiles

  Build VTK using gnu make and makefiles
- I was flabbergasted and my command prompt cursor must have been exhausted. I had never seen the little guy do so much tearing across the screen to build a 3rdParty Package. Thankfully he was still blinking at the same rate as when he started, tough little bugger.
- I vowed never to use CMake after that… Well then 2009 rolled around and the need to use VTK, ITK, dcmtk, boost etc all of witch had versions which used CMake. Quoting Homer (Simpson not Greek poet of Iliad) “D’oh!”

# What else is CMake?

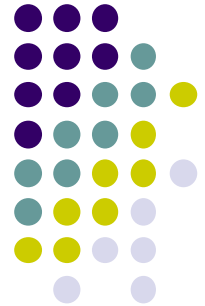
- What is meant by “**family of tools**”?
  - CTest and CDash
    - Automated test (CTest) and reporting (CDash)
    - http://www.cdash.org/CDash/index.php?project=CMake
  - CPack
    - Package software for distributions
- We will focus on CMake in this talk

# Why use CMake?



- Cross platform support
- Plain text files for build specification which can be tracked and diffed (WinMerge/Meld) easily with source control tools (git)
  - http://winmerge.org/
  - http://meldmerge.org/
- Code generation, configuration file generation, and text manipulation with configure_file(…)
- Powerful scripting
- Create your own generator
- Regular Expressions
- Not an exhaustive list!

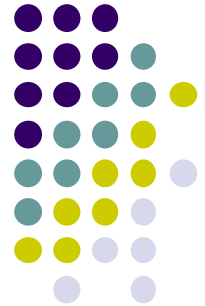
# What does CMake Look Like?



```
# The name of our project is "HELLO". CMakeLists files in this project can
# refer to the root source directory of the project as ${HELLO_SOURCE_DIR} and
# to the root binary directory of the project as ${HELLO_BINARY_DIR}.
cmake_minimum_required (VERSION 2.6)

project (HELLO)

# Recurse into the "Hello" and "Demo" subdirectories. This does not actually
# cause another CMake executable to run. The same process will walk through
# the project's entire directory structure.
add_subdirectory (Hello)
add_subdirectory (Demo)
```

Source: http://www.CMake.org/CMake/help/examples.html

# What does CMake Look Like?

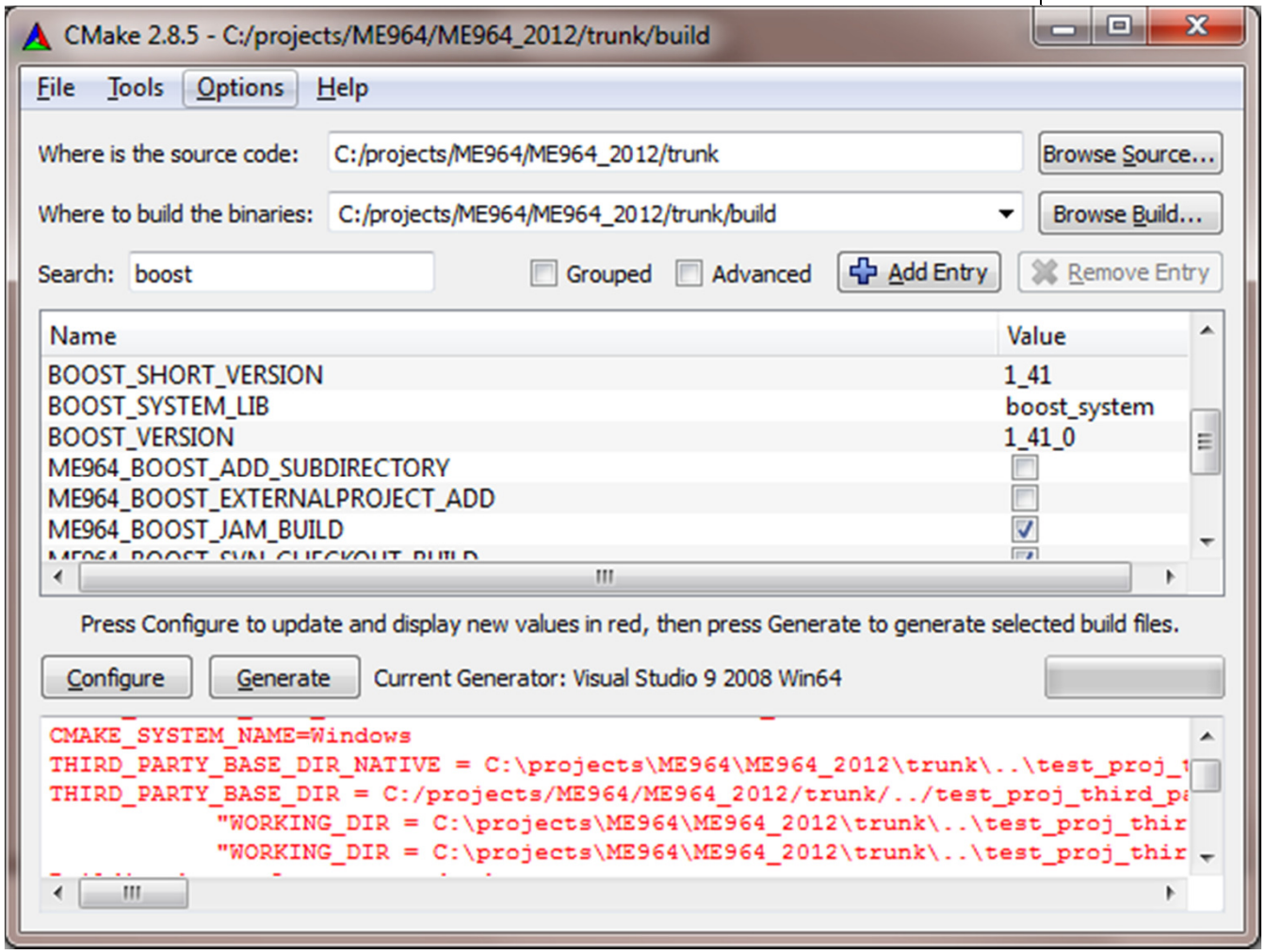


```
# Make sure the compiler can find include files from our Hello library.
include_directories (${HELLO_SOURCE_DIR}/Hello)
# Make sure the linker can find the Hello library once it is built.
link_directories (${HELLO_BINARY_DIR}/Hello)

# Add executable called "helloDemo" that is built from the source files
# "demo.cxx" and "demo_b.cxx". The extensions are automatically found.
add_executable (helloDemo demo.cxx demo_b.cxx)

# Link the executable to the Hello library.
target_link_libraries (helloDemo Hello)
```

Source: http://www.CMake.org/CMake/help/examples.html

# What does CMake Look Like?

- CMake provides a GUI – CMake GUI
- Processes root CMakeLists.txt file
  - **Not** CMakelists.txt or CMakelist.txt or CMakeList.txt or CMakelists.txt or CMakeISts.TxT – remember cross platform (Linux)
- Allows user to interact with the build and change build parameters
- Build spec creator can specify their own parameters which appear in the GUI
- Where to build the binaries – recommended out of source builds
- **Delete Cache** button was moved to File->Delete Cache in newer versions 2.6.
- When generating build specs I use copious amounts of Delete Cache and wish it were still a button (speed).

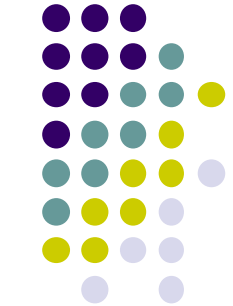



"I posted a desirement in the CMake Mantis bug tracker only to find out that all one has to do in CMake GUI is File->"Delete Cache" then config, config, generate, wait for CMake VS Macros to notice something is awry and update the projects... I still wish there were a button. If I could get a heart rate monitor to sense my level of frustration and automatically rerun a script to delete the cache I think this would be the optimal solution." – Brian J. Davis http://comments.gmane.org/gmane.comp.lib.boost.CMake/821, 2010

# CMake cache

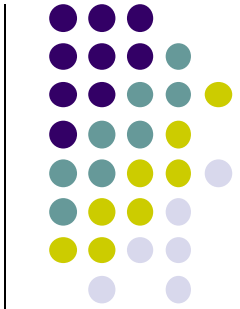
- Cache is generated when all build parameters have been set in GUI.
- Cache generation can require multiple configure steps as when build parameters change others can be activated which require user to accept or allow the user to change
- Changes are in red until accepted and new ones based on user changes will appear in red
- Clearing the cache
  - File->Delete Cache

# CMake cache



- Why a cache?
  - Speed – no need to reparse all CMakeLists.txt unless they change.
- There is a dependency on CMakeLists.txt files.
  - Consequences:
    - Do not track VS project files or Makefiles (CMake output) with source control management (SCM).
      - Show of hands, Who is Using a SCM tool? Should be everyone!!
        - Windows file explorer copy directory does not count nor does zip and copy!!!
    - VS projects and Makefiles will be regenerated.
    - Never Change CMake output!
      - Except: from my understanding you can change the cache. Though not a good idea unless careful what you change.
      - Cache is loaded when CMake is loaded to acquire previous build settings

# CMake cache



- Cache and SET(..)
  - To set variables in CMake SET(..) is used

**set(<variable> <value>**
**[[CACHE <type> <docstring> [FORCE]] | PARENT_SCOPE])**

  - Example

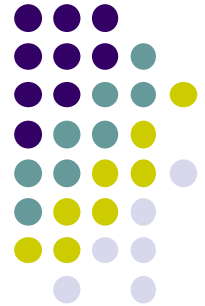
**SET( MY_STR “HELLO” CACHE STRING “hello text” FORCE )**

  - What does **FORCE** do?
    - No matter what is typed in the GUI in an attempt to change the variable the variable will always be “HELLO”
    - Be careful with FORCE as remember the cache gets reloaded when CMake is run if you decide to change a value from FORCE to not FORCED then you need to delete the cache and regenerate
  - A **non FORCE** example with use of booleans

**SET( MY_BOOL YES CACHE BOOL “my boolean” )**

    - Allows the user to change MY_BOOL in the GUI and have the modified value change in the cache
    - Can also use the preferred : **OPTION( MY_BOOL “my boolean” YES )**

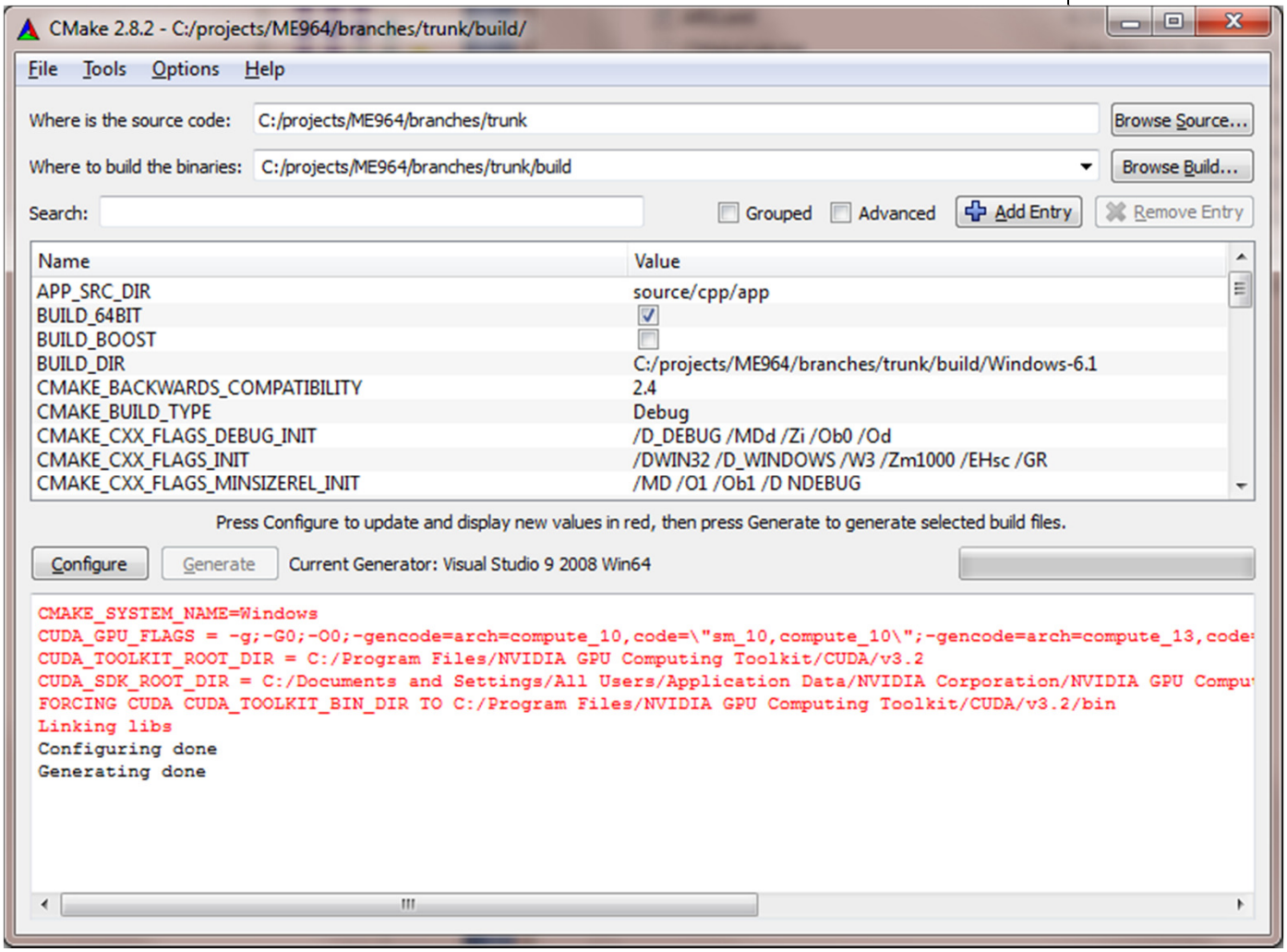
# Example Project



- Checkout using GIT
  - https://git.cae.wisc.edu/git/me964_git
- Use CAE username and password
- Run go.bat
  - Extracts CMake from zip archive – downloaded from CMake website. It is not modified.
  - Launches CMake
  - Simply for ease of use to get started
- Configure CMake variables such as
  - ME964_BUILD_BOOST
  - ME964_USE_MATLAB_INTERFACE
  - ME964_GPU_ARCH_SM10/11/13/20

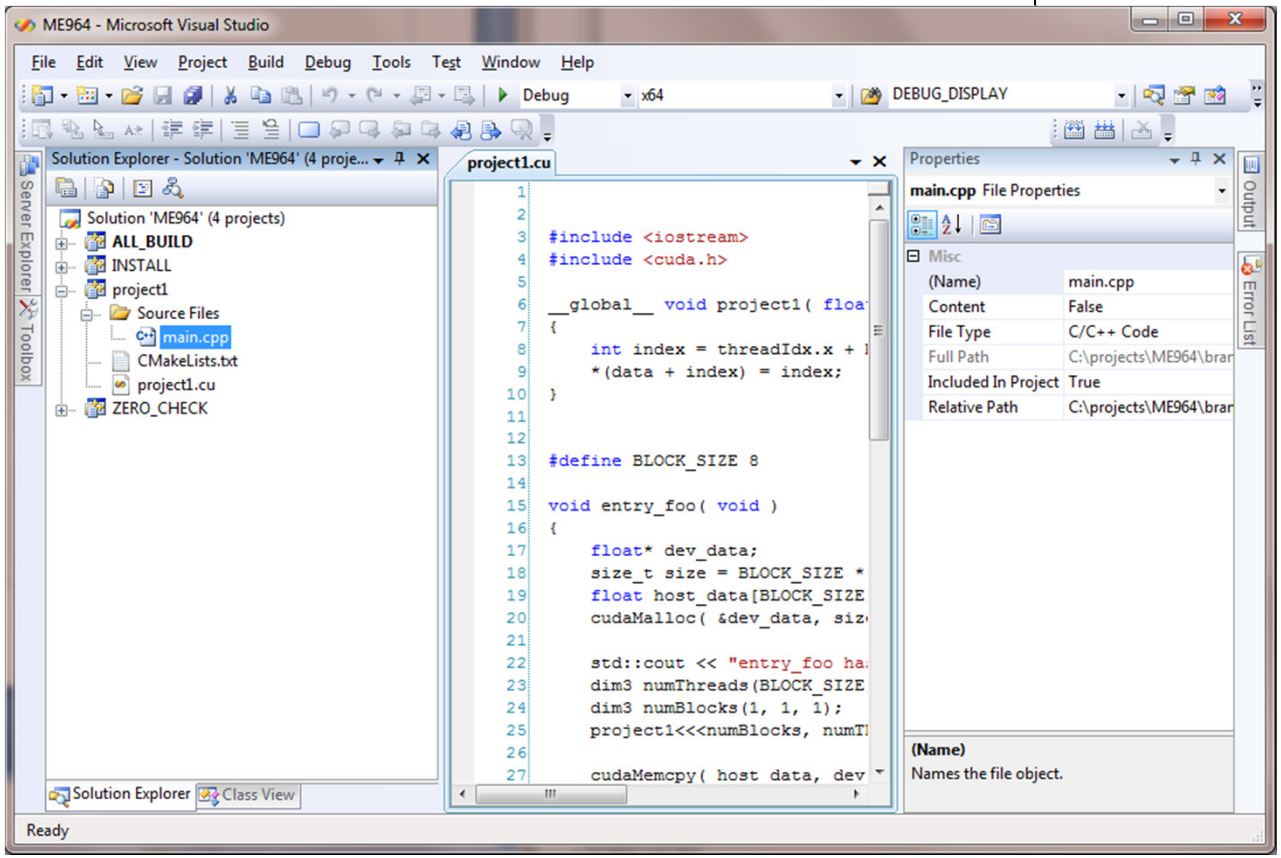
# Example Project

- Click Configure
- Click Generate
- Batch scripts are created for launching:
  - Visual Studio - run_vs.bat
  - Matlab - run_matlab.bat
  - Depwalker – run_depwalker.bat
- Batch scripts configure the PATH environment variables to ensure project DLLs are loaded from the install bin directory and so MATLAB can find GP_Loader.mex64



# Example Project

- Browse to location of top (trunk directory)
- CMakeLists.txt file then to
  - trunk\build\ME964.sln (project)
  - trunk\source\cpp\project1 (source)
- Very simple example project is generated.
- Uses SAP (my CMake code)
  - Submitted as feature request
    - http://www.CMake.org/Bug/view.php?id=11807
- **NEVER USE SEMICOLONS AFTER COMMANDS IN CMake**– can lead to all kinds of confusion as to where the error is



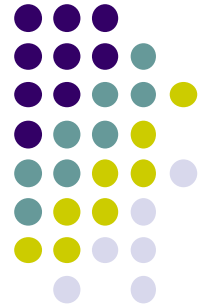

```
add_project_executable(
        # the name of your executable
        project1
        # Defines if you need any
        DEFINES
                MY_DEF=1
        # A list of .cu sources
        CU_SOURCES
                project1.cu
        # A list of .cpp sources
        CPP_SOURCES
                main.cpp
        INCLUDE_DIRECTORIES
                ../include
        INSTALL_DIRECTORIES
                bin
)
```

This is a CMake function written to simplify the common usage of creating an application. NOT part of the vanilla CMake install

# Example Project

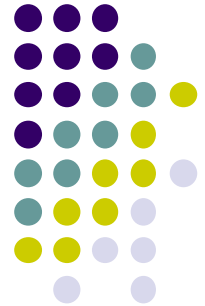
- What did I add?
  - **add_project** – not fully implemented… it's a Work in Progress (WIP) – really just a place holder for now. Goal was to have namespace resolution at least a form that would be possible in CMake by prepending variables with project name.
  - **add_project_configuration** – allows specification of a configuration which can be inherited
  - **add_project_executable** - creates a executable and can inherit project configurations (multiple)
  - **add_project_library** – creates a library which compiles C/C++/CUDA and can inherit configurations (multiple)
  - **patch** – patches files
  - **unpack** – unzips .tar, .bz2, and zip
- Remember CMake was unmodified these scripts are in the CMake project specific directory at root of tree.
- CMake build tool and related source is found in platform/3rdParty/tools directory when extracted from zip archive.

# Example Project Using Vanilla CMake



- Browse to location of **CMAKE_HOME_DIRECTORY** which is the path to top of source tree where root CMakeLists.txt file is located
- Delete cache if necessary (if you tried previous project) and changed something
- Configure and Generate project
- Browse to
  - trunk\build\ME964.sln (project)
  - trunk\source\cpp\project1_vanilla (source)
- Very simple example project is generated.

# Example Project Using Vanilla CMake



```
# Root CMakeLists.txt file

# check required version of CMake
CMAKE_MINIMUM_REQUIRED(VERSION 2.0)
#IF(CMAKE_BACKWARDS_COMPATIBILITY GREATER 2.0.6)
#  SET(CMAKE_BACKWARDS_COMPATIBILITY 2.0.6 CACHE STRING "Latest version of CMake
      when this project was released." FORCE)
#ENDIF(CMAKE_BACKWARDS_COMPATIBILITY GREATER 2.0.6)

if(COMMAND cmake_policy)
  cmake_policy(SET CMP0003 NEW)
endif(COMMAND cmake_policy)

# Declare the project
PROJECT(ME964)

SET( MY_STR “HELLO” CACHE STRING “hello text” FORCE )
SET( MY_BOOL YES CACHE BOOL “my boolean” )

add_subdirectory( source/cpp/project_vanilla/src project_vanilla )
```

# Example Project Using Vanilla CMake

- **cmake_policy**

"As CMake evolves it is sometimes necessary to change existing behavior in order to fix bugs or improve implementations of existing features. The CMake Policy mechanism is designed to help keep existing projects building as new versions of CMake introduce changes in behavior. Each new policy (behavioral change) is given an identifier of the form "CMP<NNNN>" where "<NNNN>" is an integer index. Documentation associated with each policy describes the OLD and NEW behavior and the reason the policy was introduced. Projects may set each policy to select the desired behavior. When CMake needs to know which behavior to use it checks for a setting specified by the project. **If no setting is available the OLD behavior is assumed and a warning is produced requesting that the policy be set**."

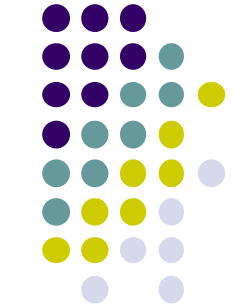
  - Want to know more then see policies section of reference listed below

- **project(<projectname> [languageName1 languageName2 ... ] )**
  - **Sets language to be used**
  - **Default C/C++**
  - **Not what I was expecting as projects can have multiple subprojects**
- **find_package(<package> [version] [EXACT] [QUIET] [[REQUIRED|COMPONENTS] [components...]] [NO_POLICY_SCOPE])**
  - **FIND_PACKAGE( CUDA )**
- **add_subdirectory(source_dir [binary_dir] [EXCLUDE_FROM_ALL])**
  - **add_subdirectory( source/cpp/project_vanilla/src project_vanilla )**
    - Adds the subdirectory source/cpp/project_vanilla/src containing a CMakeLists.txt file for the exe and library.
    - Example CMakeLists.txt file to follow.
    - EXCLUDE_FROM_ALL removes from all target  such as "make all" requiring a manual build command specifying target



Source: http://www.CMake.org/CMake/help/CMake-2-8-docs.html

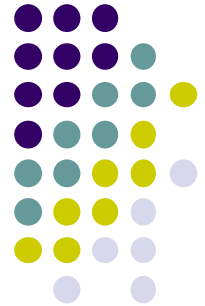
# Example Project Using Vanilla CMake



- Fairly self explanatory which is nice!
- Override where CMake would like to put the files default C:\Program Filles\ME964
- Include directories
- Need to use **CUDA_ADD_LIBRARY or CUDA_ADD_EXECUTABLE** which is from part of FindCUDA when we called **find_package( CUDA)**
- Set link flags to export function in dll
- Specify where to install the app and lib in ${CMAKE_INSTALL_PREFIX }/bin and ${CMAKE_INSTALL_PREFIX }/lib respectively
- Configure a file which uses a variable in the file throug use of syntax **${SOME_VAR}**.
  - When Configured the place holder in the file will be replaced with the value.

```
# OVERRIDE WHERE CMake WOULD LIKE TO INSTALL THE FILES
SET(
                CMAKE_INSTALL_PREFIX
                ${CMAKE_HOME_DIRECTORY}/install
                CACHE STRING "" FORCE
)

include_directories( ../include )


SET( MY_LIB_NAME libv )
SET( MY_APP_NAME project_vanilla )


# for normal C/C++ code add_library would be used,
# but since this contains a .cu file
# CUDA_ADD_LIBRARY must be used

CUDA_ADD_LIBRARY(
                ${MY_LIB_NAME}
                project_vanilla.cu
                libv.cpp
)

# need to export the function

set_target_properties(
                ${MY_LIB_NAME}
                PROPERTIES
                LINK_FLAGS /export:my_entry_function
)

add_executable(
                ${MY_APP_NAME}
                main.cpp
)

target_link_libraries( ${MY_APP_NAME}  ${MY_LIB_NAME}  )

install( TARGETS ${MY_APP_NAME}  DESTINATION bin )

install( TARGETS ${MY_LIB_NAME}  DESTINATION lib  )

SET( SOME_VAR "This is what gets put in run_program.bat when configured" )

configure_file( run_program.txt ${CMAKE_INSTALL_PREFIX}/bin/run_program.bat )
```

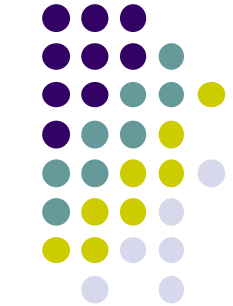
# Vanilla CMake



- Directory listing
  - If you want **find** and **grep** in Windows then install Cygwin or GNUWin32 Utils.
  - CMakeLists.txt contains both the vanilla and modified projects
  - Creates a test application and DLL called by the application

./CMakeLists.txt
./CMakeLists.txt.example
**./CMakeLists.txt.vanilla**
./go
./go.bat
./install
./install/bin
./install/bin/project_vanilla.exe
./install/bin/run_program.bat
./install/lib
./install/lib/libv.lib
./README.txt
./source
./source/cpp
./source/cpp/project_vanilla
./source/cpp/project_vanilla/include
**./source/cpp/project_vanilla/include/project_vanilla.h**
./source/cpp/project_vanilla/src
**./source/cpp/project_vanilla/src/CMakeLists.txt**
**./source/cpp/project_vanilla/src/libv.cpp**
**./source/cpp/project_vanilla/src/main.cpp**
**./source/cpp/project_vanilla/src/project_vanilla.cu**
**./source/cpp/project_vanilla/src/run_program.txt**

# Vanilla CMake Example Cont



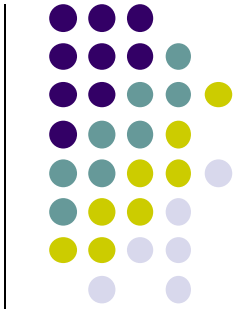
- Main.cpp (below)
- project_vanilla.cu (left)

```
#include <iostream>

#include <project_vanilla.h>

int main( void )
{
   my_entry_function();
   std::cout << "all is well in the universe\n";
}
```

```
#include <project_vanilla.h>
#include <iostream>
#include <cuda.h>

__global__ void project1( float* data )
{
    int index = threadIdx.x + blockDim.x * threadIdx.y;
    *(data + index) = index;
}


#define BLOCK_SIZE 8

_EXPORT_FUNCTION void my_entry_function( void )
{
    float* dev_data;
    size_t size = BLOCK_SIZE * BLOCK_SIZE * sizeof( float);
    float host_data[BLOCK_SIZE * BLOCK_SIZE];
    cudaMalloc( &dev_data, size );

    std::cout << "entry_foo has been entered\n";
    dim3 numThreads(BLOCK_SIZE, BLOCK_SIZE );
    dim3 numBlocks(1, 1, 1);
    project1<<<numBlocks, numThreads>>>(dev_data);

    cudaMemcpy( host_data, dev_data, size, cudaMemcpyDeviceToHost);
    for( int row = 0; row < BLOCK_SIZE; row++ )
    {
      for( int col = 0; col < BLOCK_SIZE; col++ )
      {
        std::cout << "\t" << host_data[col + row * BLOCK_SIZE];
      }
      std::cout << std::endl;

    }
    cudaFree( dev_data );
}
```

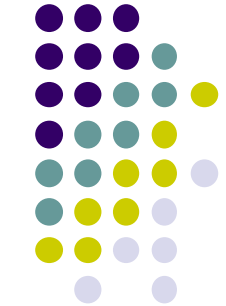
# CMake 20/80 rule?

- Along the lines of Dan's 20/80 rule
- This is more like the 80/1 rule
  - 80 percent of the functions – there aren't that many to do 1% of your total development work. This 1% of the toal work is build specification.
  - And some of these are paired like if(), else(),elseif() etc so maybe it's like the 40/1 (half as much) rule with the remaining 99 percent (of the number in denominator) going to coding in C++ which is what we should be doing anyway.
  - If you look at the example there are even fewer used, but this is a simple example. CMake allows things to get much, … much more complicated which is good.
  - Complexity when you need it simplicity when you don't

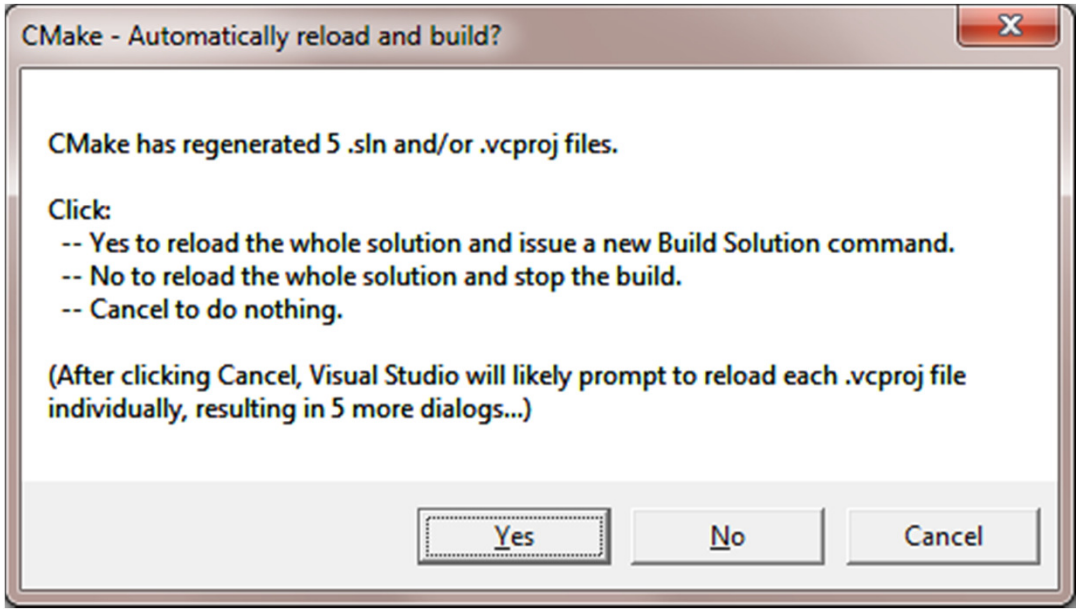


add_custom_command
add_custom_target
add_definitions
add_dependencies
add_executable
add_library
add_subdirectory
break
cmake_policy
configure_file
else
elseif
endforeach
endfunction
endif
endmacro
endwhile
execute_process
export
file
find_file
foreach
function
if
include
include_directories
install
link_directories
macro
message
option
project
return
set
string
target_link_libraries
while
add_custom_command

# CMake project regeneration



- If VS is open and CMake regenerates project files the dialog to the left will appear.
- CMake cannot be used (it is locked up) until this dialog in VS is accepted (“Yes”) and any remaining “Regenerate project” dialogs that may appear.


CMake - Automatically reload and build?
CMake has regenerated 5 .sln and/or .vcproj files.
Click:
-- Yes to reload the whole solution and issue a new Build Solution command.
-- No to reload the whole solution and stop the build.
-- Cancel to do nothing.
(After clicking Cancel, Visual Studio will likely prompt to reload each .vcproj file individually, resulting in 5 more dialogs...)
Yes
No
Cancel

# Vanilla CMake Example Cont

- run_program.txt which becomes run_program.bat after configure_file
- Note ${SOME_VAR} usage. Will be replaced with value defined in CMake when .bat file is generated
- cmd /k
  - Just keeps command window from disappearing so commands can be typed after double clicking .bat file in file explorer
- PWD sets present working directory to current directory

run_program.txt

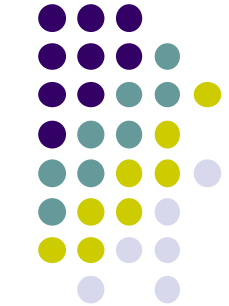
```
SET PWD=%CD%

echo ${SOME_VAR}

cmd /k
```

Run through configure_file with some_var set to SET( SOME_VAR “Hello World” )

run_program.bat

```
SET PWD=%CD%

echo Hello World

cmd /k
```

# CMake Macros/Functions and Parse Arguments



- When writing your own functions and macros you'll likely need the parse_arguments function
  - http://www.itk.org/Wiki/CMakeMacroParseArguments
  - You can probably guess what it is good for.

```
SET(arguments hello OPTION3 world LIST3 foo bar
  OPTION2 LIST1 fuz baz )

PARSE_ARGUMENTS(ARG "LIST1;LIST2;LIST3"
  "OPTION1;OPTION2;OPTION3" ${arguments})
```

- The parameters are then dereferenced with syntax ${ARG_LIST1} and ${ARG_OPTION2}  as an example

# CMake Commands

- **message**([STATUS|WARNING|AUTHOR_WARNING|FATAL_ERROR|SEND_ERROR] "message to display" ...)
  - Output shows up in Cmake GUI output window.
  - Very handy when trouble shooting build scripts

# CMake Commands

- **if() else() endif()**

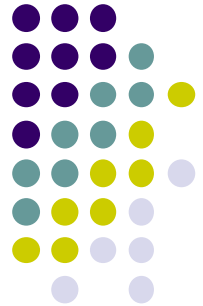
```
if(expression)
        # then section.
        COMMAND1(ARGS ...)
        COMMAND2(ARGS ...)
        ...
        elseif(expression2)
        # elseif section.
        COMMAND1(ARGS ...)
        COMMAND2(ARGS ...)
        ...
        else(expression)
        # else section.
        COMMAND1(ARGS ...)
        COMMAND2(ARGS ...)
        ...
        endif(expression)
```

- You might see this expression where beginning if and else need same expression.. I never do this and only ever put expression in first if and not in trailing else, elseif, and endif etc
- From CMake FAQ: “As of CMake 2.6.0 the ELSE() and ENDIF() constructs can be empty.”

**•There are more than thoes shown below. Just showing most commonly ones I use. Documentation is omitted see online refrence manual for more.**

if(<constant>)
if(<variable>)
if(NOT <expression>)
if(<expr1> AND <expr2>)
if(<expr1> OR <expr2>)
if(TARGET target-name)
if(EXISTS file-name)
if(EXISTS directory-name)
if(IS_DIRECTORY directory-name)
if(IS_ABSOLUTE path)
if(<variable|string> MATCHES regex)
if(<variable|string> LESS <variable|string>)
if(<variable|string> GREATER <variable|string>)
if(<variable|string> EQUAL <variable|string>)
if(<variable|string> STRLESS <variable|string>)
if(<variable|string> STRGREATER <variable|string>)
if(<variable|string> STREQUAL <variable|string>)
if(DEFINED <variable>)
if((expression) AND (expression OR (expression)))

# CMake Commands



**file(WRITE filename "message to write"... )**

**file(APPEND filename "message to write"... )**

**file(READ filename variable [LIMIT numBytes] [OFFSET offset] [HEX])**

file(STRINGS filename variable [LIMIT_COUNT num][LIMIT_INPUT numBytes] [LIMIT_OUTPUT numBytes]
[LENGTH_MINIMUM numBytes] [LENGTH_MAXIMUM numBytes]
[NEWLINE_CONSUME] [REGEX regex]
[NO_HEX_CONVERSION])

file(GLOB variable [RELATIVE path] [globbing expressions]...)

file(GLOB_RECURSE variable [RELATIVE path]
[FOLLOW_SYMLINKS] [globbing expressions]...)

**file(RENAME <oldname> <newname>)**

**file(REMOVE [file1 ...])**

file(REMOVE_RECURSE [file1 ...])

**file(MAKE_DIRECTORY [directory1 directory2 ...])**

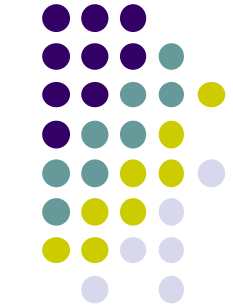
file(RELATIVE_PATH variable directory file)

**file(TO_CMAKE_PATH path result)**

**file(TO_NATIVE_PATH path result)**

file(DOWNLOAD url file [TIMEOUT timeout] [STATUS status] [LOG log] [EXPECTED_MD5 sum] [SHOW_PROGRESS])

# CMake Commands



- String regex – regular expression
- String comparisons
- To Upper/Lower case

**string(REGEX MATCH <regular_expression>**
**<output variable> <input> [<input>...])**

**string(REGEX MATCHALL <regular_expression>**
**<output variable> <input> [<input>...])**

**string(REGEX REPLACE <regular_expression>**
**<replace_expression> <output variable>**
**<input> [<input>...])**

string(REPLACE <match_string>
<replace_string> <output variable>
<input> [<input>...])

**string(COMPARE EQUAL <string1> <string2> <output variable>)**

**string(COMPARE NOTEQUAL <string1> <string2> <output variable>)**

**string(COMPARE LESS <string1> <string2> <output variable>)**

**string(COMPARE GREATER <string1> <string2> <output variable>)**

string(ASCII <number> [<number> ...] <output variable>)

string(CONFIGURE <string1> <output variable>
[@ONLY] [ESCAPE_QUOTES])

**string(TOUPPER <string1> <output variable>)**

**string(TOLOWER <string1> <output variable>)**

string(LENGTH <string> <output variable>)

string(SUBSTRING <string> <begin> <length> <output variable>)

string(STRIP <string> <output variable>)

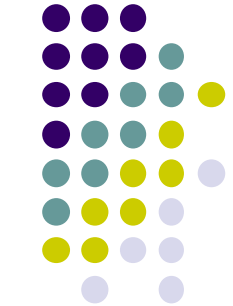
string(RANDOM [LENGTH <length>] [ALPHABET <alphabet>]
[RANDOM_SEED <seed>] <output variable>)

# ExternalProject_ADD

- Adds external 3rdParty packages to your projects
  - Can download source from repositories: SVN, GIT, CVS
  - Build and install source that it CMake friendly
  - There is an example though not currently working (keeps rebuilding itself) using boost and svn which can be activated with **ME964_BOOST_SVN_CHECKOUT_BUILD** for the curious
- Once it was elusive and it's documentation could only be accessed by the command prompt.

```
ExternalProject_Add(<name> # Name for custom target
[DEPENDS projects...] # Targets on which the project depends
[PREFIX dir] # Root dir for entire project
[LIST_SEPARATOR sep] # Sep to be replaced by ; in cmd lines
[TMP_DIR dir] # Directory to store temporary files
[STAMP_DIR dir] # Directory to store step timestamps
#--Download step--------------
[DOWNLOAD_DIR dir] # Directory to store downloaded files
[DOWNLOAD_COMMAND cmd...] # Command to download source tree
[CVS_REPOSITORY cvsroot] # CVSROOT of CVS repository
[CVS_MODULE mod] # Module to checkout from CVS repo
[CVS_TAG tag] # Tag to checkout from CVS repo
[SVN_REPOSITORY url] # URL of Subversion repo
[SVN_REVISION rev] # Revision to checkout from Subversion repo
[SVN_USERNAME john ] # Username for Subversion checkout and update
[SVN_PASSWORD doe ] # Password for Subversion checkout and update
[GIT_REPOSITORY url] # URL of git repo
[GIT_TAG tag] # Git branch name, commit id or tag
[URL /.../src.tgz] # Full path or URL of source
[URL_MD5 md5] # MD5 checksum of file at URL
[TIMEOUT seconds] # Time allowed for file download operations
#--Update/Patch step----------
[UPDATE_COMMAND cmd...] # Source work-tree update command
[PATCH_COMMAND cmd...] # Command to patch downloaded source
#--Configure step-------------
[SOURCE_DIR dir] # Source dir to be used for build
[CONFIGURE_COMMAND cmd...] # Build tree configuration command
[CMAKE_COMMAND /.../CMake] # Specify alternative CMake executable
[CMAKE_GENERATOR gen] # Specify generator for native build
[CMAKE_ARGS args...] # Arguments to CMake command line
[CMAKE_CACHE_ARGS args...] # Initial cache arguments, of the form -Dvar:string=on
#--Build step-----------------
[BINARY_DIR dir] # Specify build dir location
[BUILD_COMMAND cmd...] # Command to drive the native build
[BUILD_IN_SOURCE 1] # Use source dir for build dir
#--Install step---------------
[INSTALL_DIR dir] # Installation prefix
[INSTALL_COMMAND cmd...] # Command to drive install after build
#--Test step------------------
[TEST_BEFORE_INSTALL 1] # Add test step executed before install step
[TEST_AFTER_INSTALL 1] # Add test step executed after install step
[TEST_COMMAND cmd...] # Command to drive test
#--Output logging-------------
[LOG_DOWNLOAD 1] # Wrap download in script to log output
[LOG_UPDATE 1] # Wrap update in script to log output
[LOG_CONFIGURE 1] # Wrap configure in script to log output
[LOG_BUILD 1] # Wrap build in script to log output
[LOG_TEST 1] # Wrap test in script to log output
[LOG_INSTALL 1] # Wrap install in script to log output
#--Custom targets-------------
[STEP_TARGETS st1 st2 ...] # Generate custom targets for these steps
)
```

# Some points about CMake



- Flexible and powerful through use of a few commands
- There are more CMAKE_ variables see doc for more info
- Build settings are local to the directory.
  - What does this mean?
    - Build settings specified are “Global” to the directory unless set_target_properties is used
    - set_target_properties useful when you have 2 libs which use same source file with #ifdefs and you want to build them with different settings.
- Settings can have build configuration specific settings such as
  - LINK_FLAGS_<CONFIG>
  - LINK_FLAGS_DEBUG
  - LINK_FLAGS_RELEASE
- Beware of CACHE FORCE and clear cache if you think something is not quite right.
- IMO - Needs the concept of namespaces as CMake variables can grow in large projects
- project() is not what I expected
- add_project – does not work for third party packages such as vtk, dcmtk, ITK etc and ExternalProject_Add must be used.
  - This IMO does not allow to build for only what you need. - No true dependency checking across files. What gets build is all or none.
- Use fully qualified names for paths NOT relative.
  - Try relative first, but when it doesn't work switch to fully qualified names.
- Tar was supported in CMake –E, but not zip – used for unpacking source zip and tarballs.
- Watch for deprecated functions. Mostly what I see is consolidation (consolidating functions into one and increasing parameters) which makes sense.

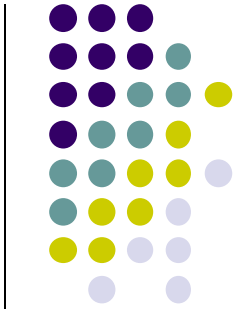
# MATLAB / C++ interface

- The example project provides a MATLAB / C++ interface.
- Activate by enabling **ME964_USE_MATLAB_INTERFACE** by using search bar in Cmake GUI
- Uses a template class mex_matrix<T>
  - **mex_matrix<float> data_matrix( &prhs[MATRIX_DATA_INDEX] );**
- #include <mexlib/mex_matrix.h>
- Example simply copies a source array to a destination array however by using the **T* get_ptr()** function copying data to device memory is straight forward.

# MATLAB / C++ interface

- Including the full cuda/matlab test example would have pulled in too many dependencies from my code base so I have the example leaving off at the MATLAB / C layer
- mex_c_interface.m in the directory source\Matlab\test_cases\mex_c_interface contains the matlab test code

# MATLAB / C++ interface

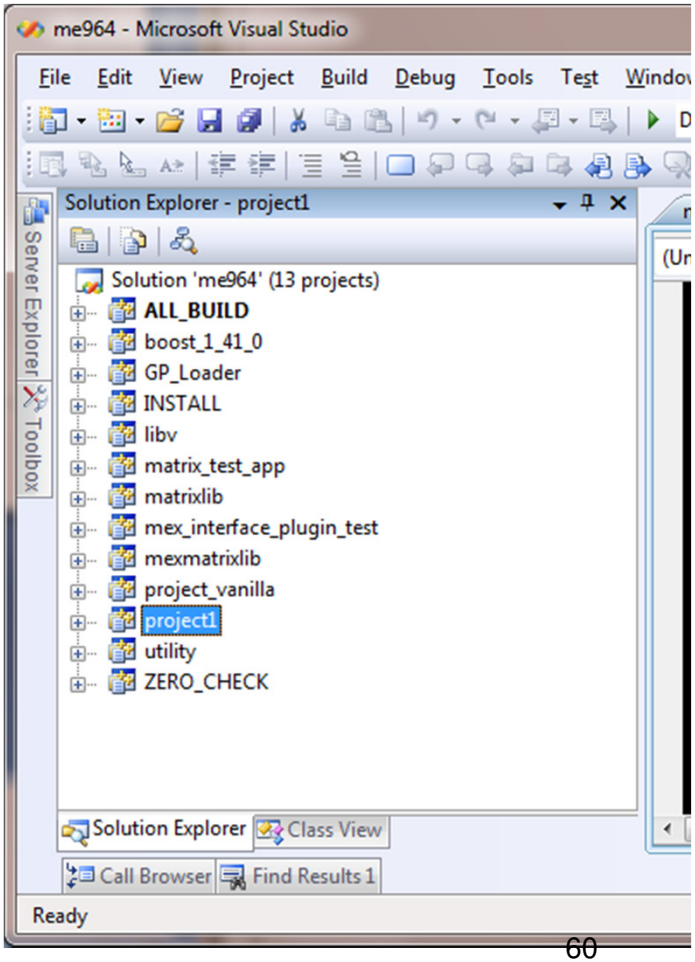


- Sets the mex plugin to be loaded by GP_Loader
- **GP_Loader** allows for a generic dll loader to call the mex_function and **allows swapping of dlls out from under Matlab** so long as GP_Loader is set to build last

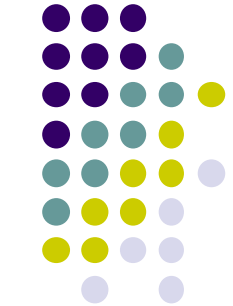
```
addpath(genpath( '../../../../' ))
%%
x_dim = 4
y_dim = 5
z_dim = 2
num_elements = x_dim * y_dim * z_dim;

array = 1:num_elements
array = reshape(array, [x_dim  y_dim  z_dim] )
orig_array = array
size(array)
%%
dims = size(orig_array)
%%
clc
array = single( array )

plugin = 'mex_interface_plugin_test';

ret_array = GP_Loader( plugin, array );

ret_array == array

size( orig_array )
size( ret_array )

if( isempty( find((ret_array ~= array)) ) )
   disp( 'TEST PASSED')
else
   disp( 'TEST FAILED')
end
```

# MATLAB / C++ interface

- Enabeling ME964_USE_MATLAB_INTERFACE in CMake will generate projects
  - boost_1_41_0
  - GP_Loader
  - matrix_test_app
  - matrixlib
  - mex_interface_plugin_test
  - mexmatrixlib
  - utility



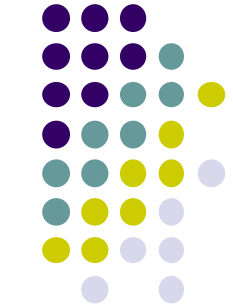
# MATLAB / C++ interface



- **mex_function** is entry point that GP_Loader seeks to find.
- Note that the standard MATLAB function is **mexFunction** as it is inside GP_Loader
- Shows use of print_array(), print_dimensions() and get_data_ptr().
- Print calls use **LOG_EVENT** and are written to log file as defined by log_file_name = "**me964.log**"; in **event_logger.cpp** as built in the **utility** library
- Use **mxReal** for real data and **mxComplex** for complex data. Complex data was tested with FFT library so this should work.
- Here data_t is: typedef float data_t;

```
/* Gateway function */
#ifdef __cplusplus
extern "C" {
#pragma message( "building for C++" )
#endif
void mex_function
(
                        int nlhs, mxArray *plhs[],
                        int nrhs, const mxArray *prhs[]
)
{
try {
      // Check if there are the correct number of arguments
      if( !check_number_matricies( NUM_ARGS, nrhs ) ) return;

       // create a matrix object for the projection data
      mex_matrix<data_t> data_matrix( &prhs[MATRIX_DATA_INDEX] );


      LOG_EVENT( LOG_STATUS, "matrix dimensions\n" );
      data_matrix.print_dimensions();

      data_matrix.print_array();


      if( nrhs > 0 ){
          mex_matrix<data_t> ret_data_matrix( &plhs[0], data_matrix.get_dimensions(), mxREAL );
          memcpy( ret_data_matrix.get_data_ptr(), data_matrix.get_data_ptr(),
ret_data_matrix.get_size_in_bytes() );
                                                    }

      }
catch( std::exception& e )
{
        LOG_EVENT( LOG_ERROR,"Error: %s\n", e.what() );
}
    LOG_EVENT( LOG_STATUS, "==================== TEST END ======================\n" );

}

#ifdef __cplusplus
}
#endif
```

# CUDA Debugging



- Bug types
- Why use a debugger?
- Where did printf go?
- Current state of tools
- NSight
- cuda-gdb
- OpenCL

"I might at this point be better off programming an 256x256x256 array of industrial robots to move around beads on a matched series of abacuses. At least I could physically see where the problem was occurring." -- Brian J. Davis, NVIDIA Developer Zone posting: [NSIGHT Confused by ? shows ??? I know I am confused] - 2011

# Bug types

- **Bohr bug** – "A repeatable *bug*; one that manifests reliably under a possibly unknown but well-defined set of conditions. Antonym of *heisenbug*;"
- **Heisenbug** - "A bug that disappears or alters its behavior when one attempts to probe or isolate it. (This usage is not even particularly fanciful; the use of a debugger sometimes alters a program's operating environment significantly enough that buggy code, such as that which relies on the values of uninitialized memory, behaves quite differently.) Antonym of *Bohr bug*"
- **Mandelbug** (Mandelbrot) – "A bug whose underlying causes are so complex and obscure as to make its behavior appear chaotic or even non-deterministic."
- **Schroedinbug** (Schroedinger's Cat thought-experiment) – "A design or implementation bug in a program that doesn't manifest until someone reading source or using the program in an unusual way notices that it never should have worked, at which point the program promptly stops working for everybody until fixed. Though (like *bit rot*) this sounds impossible, it happens; some programs have harbored latent schroedinbugs for years."
- **Phase of the Moon bug** – "The phase of the moon is sometimes spouted as a silly parameter on which a bug might depend, such as when exasperated after trying to isolate the true cause. The Jargon File documents two rare instances in which data processing problems were actually caused by phase-of-the-moon timing." Think Y2K. **Yes computers do manifest certain weird behavior based on the alignment of the planets**
- **Statistical (Stat) bug** – "Statistical bugs can only be detected in aggregates and not in single runs of a section of code. These are bugs that usually affect code that is supposed to produce random or pseudo-random output."
- Source:
  - http://en.wikipedia.org/wiki/Jargon_File
  - Jargon File - http://www.catb.org/jargon/
  - http://www.dourish.com/goodies/jargon.html
  - The Meaning of hack: http://www.catb.org/jargon/html/meaning-of-hack.html

# Why use a debugger?

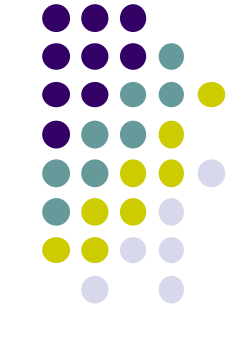
- Clean code – no need to sprinkle with printf, #ifdef _DEBUG… #endif, or macros
- Zero in on the thread, block, and grid (ID) that is causing the problem and see state of variables.
- Set watch points and if equals, less, and more (boolean) operations etc
- If it (Language) does not have a debugger in this day and age I don't use it.  i.e. I don't waste my time (except for playing Angry Birds, but that's a choice).
- SAVE TIME! SAVE TIME! SAVE TIME! That can be better utilized playing Angry Birds or sailing.

# Where did printf go?

- Removed prior to 3.0? (not sure exactly) release
  - NVIDIA CUDA Linux Release Notes Version 3.1
    - Added the ability to call printf() from kernels. This feature is supported only on the Fermi architecture.

- Even a better question why printf in the first place?
  - Let's think about this for a second.
  - Which thread in what block int the grid is this printf running?
    - When does it run?
      - Might need it in he future so add a #define #ifdef #endif. Result = messy code
  - Without if statements this can generate a lot of text.
  - How do I read this text? This text has to be copied over to the CPU.
  - Who's going to read all that text?
  - Even armed with grep… not me.
  - Should be optimizing GPU code for calculation throughput not printf statements

# Current Tools



- From NVIDIA's Website : http://developer.nvidia.com/tools-ecosystem#Debugging
- There are likely more as this is not meant to be an exhaustive list and discussion of every debugger
- This talk will focus on 2
  - Parallel Nsight
  - cuda-gdb
- Visual Profiler was covered in a previous talk

DEBUGGING & PROFILING

| COMPANY/ORGANIZATION | PRODUCT | BRIEF DESCRIPTION |
|---|---|---|
| Allinea | DDT | A single tool that can debug hybrid MPI, OpenMP and CUDA applications on a single workstation or GPU cluster. |
| Rogue Wave | TotalView | Allows Linux X86-64 users to debug both the CPU and the GPU code in CUDA applications, using familiar TotalView GUI methods. |
| NVIDIA | CUDA-GDB | The CUDA-GDB debugger is an extended version of the open source gdb debugger, providing seamless native hardware debugging across CPUs and GPUs. |
| NVIDIA | Parallel Nsight | NVIDIA® Parallel Nsight™ is a free powerful plugin that allows programmers to develop for both GPUs and CPUs within Microsoft Visual Studio. |
| NVIDIA | Visual Profiler | Visual Profiler is a cross-platform performance profiling tool that delivers developers vital feedback for optimizing CUDA C/C++ and OpenCL applications. |
| NVIDIA | CUDA-MEMCHECK | CUDA-MEMCHECK checks for misaligned memory, out of bounds addresses and many other application errors. |

# Parallel Nsight

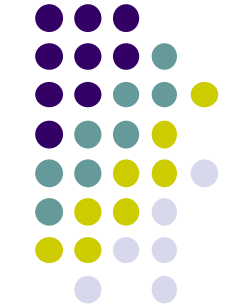
- CUDA C/C++ Debugging
- CUDA Kernel Trace/Profiling
- Data breakpoints for CUDA C/C++ code
- OpenCL Kernel Trace/Profiling – OK but what about debugging?
- Now with VS 2010 support

## Parallel Nsight - Features

All NVIDIA® Parallel Nsight™ Professional features are now available free of charge in a no cost download.

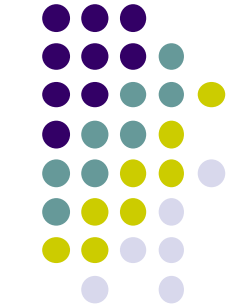
| Feature | No-cost Download |
|---|---|
| Integrated into Visual Studio 2008 SP1 or Visual Studio 2010 | ✓ |
| CUDA C/C++ Debugging | ✓ |
| DirectX 10/11 Shader Debugging | ✓ |
| DirectX 10/11 Frame Debugging | ✓ |
| DirectX 10/11 Frame Profiling | ✓ |
| CUDA Kernel Trace/Profiling | ✓ |
| OpenCL Kernel Trace/Profiling | ✓ |
| DirectX 10/11 API & HW Trace | ✓ |
| Data breakpoints for CUDA C/C++ code | ✓ |
| Analyzer/System Trace | ✓ |
| Tesla Compute Cluster (TCC) Support | ✓ |
| Forum Support | ✓ |
| All Version Upgrades | ✓ |

# Nsight Monitor and Debugger



- Nsight monitor must be started before the program to be debugged can be launched
- Nsight debugger attaches to monitor
- Secured Connections
  - Allows only certain computers to connect
- File synchronization
  - Needed if remote debugging and dlls, config files, etc are needed.

# Nsight Configuration



- Nsight Monitor
  - Headed Mode - now called Local Mode
    - Can be used when computer has more than 1 GPUs installed
    - WDDM TDR
      - “TDR stands for Timeout Detection and Recovery. This is a feature of the Windows operating system which detects response problems from a graphics card, and recovers to a functional desktop by resetting the card. If the operating system does not receive a response from a graphics card within a certain amount of time (default is 2 seconds), the operating system resets the graphics card. Before TDR existed, problems of this nature would have resulted in a system freeze and required a reboot of the operating system. If TDR is enabled and you see the TDR error message "Display driver stopped responding and has recovered", this means that the Windows operating system reset the display driver.” – refrence Nsight User manual
      - TDR crashes will also be seen on long running kernels on single gpu computers
  - Headless Mode - now called Remote Mode
    - Used when there is no display
    - Connections are made remotely from client running Nsight – I have not tried this… yet as I run headed mode with multiple GPUs

# Nsight Requirements

- From: http://http.developer.nvidia.com/ParallelNsight/1.51/UserGuide/HTML/webframe.html
  - OS
    - Windows® Vista (32 or 64-bit) with SP1, or
    - Windows® 7 (32 or 64-bit), or
    - Windows HPC Server 2008 (32 or 64-bit)
  - Local debugging (Headed Mode) (host and target on same machine)
    - 2 GPUs, each must be either a G92, GT200, or GT400 GPU. See below for supported graphics cards.
  - Remote debugging (Headless Mode) (host and target on different machines)
    - On the target machine:
      - 1 GPU on target machine: must be a G92, GT200, or GT400 GPU*
    - On the host machine (with Visual Studio):
      - 1 GPU on host machine: can be any GPU
  - Current supported cards at left

| GeForce | Quadro | Tesla |
|---|---|---|
| GeForce GTX 580 | Quadro 6000 | Tesla C2050 |
| GeForce GTX 570 | Quadro 5000 | Tesla C2070 |
| GeForce GTX 560 | Quadro 4000 | Tesla C1060 |
| GeForce GTX 480 | Quadro FX 4800 | Tesla S1070 |
| GeForce GTX 470 | Quadro CX | |
| GeForce GTX 460 | Quadro FX 5800 | |
| ION | Quadro FX 3800 | |
| GeForce GTX 260 | Quadro FX 3700 | |
| GeForce GTX 260 | Quadro FX 1800 | |
| GeForce GTX 275 | Quadro FX 380 | |
| GeForce GTX 280 | Quadro FX 580 | |
| GeForce GTX 285 | Quadro FX 370 | |
| GeForce GTX 295 | Quadro FX 380 LP | |
| GeForce 8800 GS | | |
| GeForce 9600 GSO | | |
| GeForce 8800 GT | | |
| GeForce 9800 GT | | |
| GeForce GTS 240 | | |
| GeForce 8800 GTS 512 | | |
| GeForce 9800 GTX | | |
| GeForce 9800 GTX+ | | |
| GeForce GTS 250 | | |
| GeForce 9800 GX2 | | |
| GeForce 9600 GS | | |
| GeForce 9600 GSO 512 | | |
| GeForce 9600 GT | | |
| GeForce G210 | | |
| GeForce 9400 GT | | |
| GeForce 9500 GS | | |
| GeForce GT 120 | | |
| GeForce 9500 GT | | |
| GeForce G100 | | |
| GeForce 9300 GE | | |
| GeForce 8400 | | |
| GeForce 9300 SE | | |
| GeForce 8400 GS | | |
| GeForce 9300 GS | | |
| GeForce GT 220 | | |
| GeForce NVS 2100 | | |
| GeForce G210 | | |
| GeForce NVS 3100 | | |
| GeForce 210 | | |
| GeForce 310 | | |

# Nsight Requirements

- From: http://http.developer.nvidia.com/ParallelNsight/1.51/UserGuide/HTML/webframe.html
  - Disable D3D acceleration for WPF (applies to local debugging only) Open Windows Explorer.
    - Browse to the Common folder:
      - On a Windows 32-bit system browse to: C:\Program Files\NVIDIA Parallel Nsight 1.51\Common
      - On a Windows 64-bit system browse to: C:\Program Files (x86)\NVIDIA Parallel Nsight 1.51\Common
  - Double-click on the file named:
    - DisableWpfHardwareAcceleration.reg
- Known working from experience:
  - Foxconn destroyer motherboard with onboard NVIDIA® 780a SLI Chipset and 4 Tesla c1060s works just fine.
  - Also now Remote Desktop debugging works without device enumeration issue.
    - Headed GPU is not enumerated when running remote

| GeForce | Quadro | Tesla |
|---|---|---|
| GeForce GTX 580 | Quadro 6000 | Tesla C2050 |
| GeForce GTX 570 | Quadro 5000 | Tesla C2070 |
| GeForce GTX 560 | Quadro 4000 | Tesla C1060 |
| GeForce GTX 480 | Quadro FX 4800 | Tesla S1070 |
| GeForce GTX 470 | Quadro CX | |
| GeForce GTX 460 | Quadro FX 5800 | |
| ION | Quadro FX 3800 | |
| GeForce GTX 260 | Quadro FX 3700 | |
| GeForce GTX 260 | Quadro FX 1800 | |
| GeForce GTX 275 | Quadro FX 380 | |
| GeForce GTX 280 | Quadro FX 580 | |
| GeForce GTX 285 | Quadro FX 370 | |
| GeForce GTX 295 | Quadro FX 380 LP | |
| GeForce 8800 GS | | |
| GeForce 9600 GSO | | |
| GeForce 8800 GT | | |
| GeForce 9800 GT | | |
| GeForce GTS 240 | | |
| GeForce 8800 GTS 512 | | |
| GeForce 9800 GTX | | |
| GeForce 9800 GTX+ | | |
| GeForce GTS 250 | | |
| GeForce 9800 GX2 | | |
| GeForce 9600 GS | | |
| GeForce 9600 GSO 512 | | |
| GeForce 9600 GT | | |
| GeForce G210 | | |
| GeForce 9400 GT | | |
| GeForce 9500 GS | | |
| GeForce GT 120 | | |
| GeForce 9500 GT | | |
| GeForce G100 | | |
| GeForce 9300 GE | | |
| GeForce 8400 | | |
| GeForce 9300 SE | | |
| GeForce 8400 GS | | |
| GeForce 9300 GS | | |
| GeForce GT 220 | | |
| GeForce NVS 2100 | | |
| GeForce G210 | | |
| GeForce NVS 3100 | | |
| GeForce 210 | | |
| GeForce 310 | | |

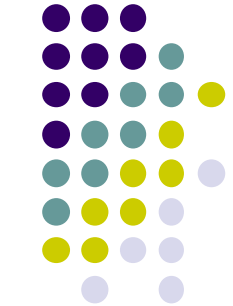
# Nsight in Action - Debugger

http://developer.nvidia.com/parallel-nsight-videos

# Nsight in Action - Profiler

http://developer.nvidia.com/parallel-nsight-videos

# cuda-gdb



- More evolved than Nsight - though v2.0 is promising
- Released before Nsight
- DDD with cuda-gdb
  - ddd --debugger cuda-gdb
- Emacs – below text is from the url at the bottom of the slide.

CUDA-GBD works with GUD in Emacs and XEmacs . No extra step is required besides

pointing to the right binary.

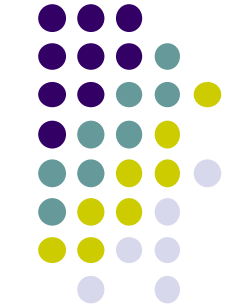
To use cuda-gdb, the 'gud-gdb-command-name' variable must be set to "cuda-gdb --

annotate=3". Use M-x customize-variable to set the variable.

Ensure that cuda-gdb is present in the Emacs/XEmacs $PATH.

http://developer.download.nvidia.com/compute/cuda/4_0_rc2/toolkit/docs/cuda-gdb.pdf pg 16

# cuda-gdb



- Remote ssh – for instance I remote debug my single gpu on my laptop using a remote ssh session
- I have had success using this method
  - ssh –X usernam@wherever
  - Stop your display manager (Ubuntu 10.10 x64 cmd shown below using gnome)
    - sudo stop gdm
    - set PATH and LD_LIBARY_PATH as necessary
    - ddd –debugger cuda-gdb app_name

# OpenCL – er what?

- **Q: Is there a OpenCL debugger?**
  There is **some** support for OpenCL in Parallel Nsight
- Remember what I said.  If it doesn't have a debugger I don't use it.
- When OpenCL does then I'll switch.

http://developer.nvidia.com/cuda-faq